НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«До захисту допущено»
Завідувач кафедри
_____ __*Павлов О.А.*
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмне забезпечення систем»*

на тему: «Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства»

Виконав: студент 4 курсу, групи ІП-51

Гарбовський Максим Вікторович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доц. к.т.н. Смаковський Д.С.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *ст.вик. Головченко М.М.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут ім. І.Сікорського”

Факультет (інститут) Інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматизованих систем обробки інформації і управління
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Павлов О.А.*
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Гарбовський Максим Вікторович
(прізвище, ім’я, по батькові)

1. Тема проекту *«Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства»*

керівник проекту Смаковський Д.С., доц. к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету від « » 2019 р. №

2. Термін подання студентом проекту « » *2019 року*

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1) Моделювання та конструювання програмного забезпепечення
2) Аналіз якості та тестування програмного забе зпечення
3) Впровадження та супровід програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових кресленників, плакатів, презентацій тощо)

1) Схема структурна класів програмного забезпечення

2) Схема структурна діяльності

3) Креслення вигляду звітних форм

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання « » *2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	*Вивчення предметної області*	21.04.2019	
2	*Аналіз існуючих методів розв'язання задачі*	25.04.2019	
3	*Постановка та формалізація задачі*	1.05.2019	
4	*Аналіз вимог до програмного забезпечення*	9.05.2019	
5	*Моделювання програмного забезпечення*	18.05.2019	
6	*Оформлення пояснювальної записки*	24.05.2019	
7	*Подання ДП на попередній захист*	28.05.2019	
8	*Подання ДП рецензенту*	01.06.2019	
9	*Подання ДП на основний захист*	08.06.2019	

Студент _____ Гарбовський М.В.
(підпис)

Керівник проекту _____ Смаковський Д.С.
(підпис)

АНОТАЦІЯ

Пояснювальна записка викладена на 71 сторінці, містить 4 розділи, 14 рисунків, 29 таблиць та 8 джерел.

Метою проекту є розробка програмного забезпечення, що дозволило б фермерам автоматизувати процес контролю, представлення та аналізу серії вимірів здійснених пенетрометром про щільність грунту в різних просторових точках поля. Що також передбачає генерацію репортів у різних форматах, таких як xml, pdf та табличного документу.

У першому розділі представлені основні поняття про предметну область, проаналізовано існуючі технічні рішення та успішні програмні продукти.

У розділі «Моделювання та конструювання програмного забезпечення» наведено схеми бізнес процесів та розроблено архітектуру платформи та окремого сервісу. Також були описані використані інструменти та засоби розробки, підтримки та розгортання коду та їх переваги при розробці програмного забезпечення даного типу.

Третій розділ містить опис підходу до тестування, методи та об’єкти тестування та основні сценарії тестування.

У розділі «Впровадження та супровід програмного забезпечення» описано процес розгортання програмного забезпечення за допомогою засобів автоматизації.

Ключові слова: ГЕОІНФОРМАЦІЙНА СИСТЕМА, МІКРОСЕРВІСИ, ГЕОДАНІ, ВЕБ-ДОДАТКИ

.

ABSTRACT

The explanatory note is set out on 71 pages, containing 4 sections, 14 figures, 29 tables and 8 sources.

The goal of the project is to develop software that would enable farmers to automate the process of control, presentation and analysis of a series of measurements made by the penetrometer on the density of soil in various spatial points of the field. It also provides the generation of reports in various formats, such as xml, pdf, and spreadsheet.

The first section presents the basic concepts of the subject area, analyzes the existing technical solutions and successful software products.

In the "Modeling and Designing Software" section the business process diagrams are presented and the architecture of the platform and a separate service is developed. The tools and tools used to develop, maintain and deploy code and their benefits in software development of this type were also described.

The third section contains a description of the approach to testing, the methods and test objects, and the main testing scenarios.

The "Software Implementation and Maintenance" section describes how to deploy software using automation tools.

Key words: GIS, MICROSERVICES, GEODATA, WEB-APPLICATIONS.

Пояснювальна записка
до дипломного проекту

на тему: «Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства»

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГІС - геоінформаційна система

SOA - Сервіс-орієнтована архітектура

Spatio-temporal location - просторово-часове розташування

MOSS - накладення карт і статистична система

GRASS GIS - Система підтримки аналізу географічних ресурсів

GPS - система глобального позиціонування

MSA - архітектура мікросервісу

BPMN – система умовних позначень для моделювання бізнес-процесів

ВСТУП

Інформаційні системи, що виникли у середині 20 століття повністю та назавжди змінив ведення бізнесу. З часом їх вплив поширився і на більш традиційні його галузі. Зараз передові компанії, що створюють апаратне та програмне забезпечення, надають комплексні рішення для великих сільськогосподарських підприємств, що полегшують автоматизовують процеси пов'язані з висадкою культур, доглядом за ними протягом періоду дозрівання та під час збору врожаю. Також вони беруть велику частину роботи, пов'язаною з моніторингом середовища (грунт, вода, повітря), аналізом історичних даних та навіть надають прогнози та рекомендації на майбутні періоди.

Ці заходи важливі для більш ефективного використання обмежених земельних ресурсів для задоволення попиту продуктів постійно зростаючого населення.

І хоча великі інформаційні системи вперше застосовуються та перевіряються великими аграрними підприємствами, розробка рішень для малого та середнього сільського господарства зробить ще більший прорив, так як саме вони є виробником більшої частини харчової сировини для сільського господарства.

Одна з характеристик грунту - щільність - має безпосередньо вплив на урожайність, здоров'я грунту та культур, інтенсивності врожайності та правильним рухом повітря та води у верхніх шарах грунту, звідки рослини беруть ресурси для зростання. Розробка геоінформаційної системи для щільності грунту дозволить більш точно та систематично виявляти проблеми та дасть великий набір даних для побудови стратегії для їх рішення.

Дана дипломна робота присвячена розробці мікросервісу генерації звітів про виміри на певному полі і стане частиною геоінформаційної системи для дослідження стану грунту через аналіз його щільності. Звіти дозволяють використовувати дані як у фінальному вигляді, так і як агреговані вхідні дані

для побудови більш складніших моделей стану грунту від декількох змінних. Крім того, рішення можна легко розгорнути як на окремий власний сервер, так і на хмарну серверну інфраструктуру і інтегрувати його як компонент більш складної геоінформаційної системи.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Геоінформаційна система (ГІС) це інструмент для створення візуальних представлень даних та проведення просторового аналізу з метою здійснення освічених рішень. Це технологія, що комбінує апаратне забезпечення, програмне забезпечення та дані. Дані можуть представляти практично що завгодно, до тих пір, поки вони мають географічну сладову. Апаратним забезпеченням може бути будь-що від персонального комп'ютера, до супутників, дронів чи переносних GPS пристроїв. Існують різні пакети програмних продуктів для подальшої обробки та аналізу зібраних даних під різні види даних.

Під час просторового аналізу можна порівнювати різні змінні, такі як тип грунту, напрямок вітру, кількість опадів, нахил чи підйом для допомоги з управління насінням, придатністю ділянок та планування дренажу, так само як усунення ризику повеней, посух, ерозії та захворювань. ГІС можуть допомогти фермерам адаптуватися до цих багатьох змінних, відслідковувати стан здоров'я окремого посіву, оцінювати врожай з певного поля. Існує багато джерел для ГІС даних як безкоштовних, так і платних. Університети, уряди країн, агенції та приватні компанії - всі вони є сховищами просторових даних. ГІС є також сприяє зусиллям, спрямованим на усунення глобального голоду.

Супутники, дрони, пілотовані літальні засоби використовуються для віддаленого зондування, що збирає інформацію про стан земної поверхні шляхом сканування її з великої висоти. Так, наприклад, NASA запустила супутник спостереження, що обертається навколо Землі кожні 16 днів. Він захоплює 9 смуг видимого світлового спектру, що можуть бути використаними для обчислення коефіцієнтів про хвороби рослин, недоліку поживних речовин, зараження комахами, надлишок або дефіцит вологості рослин. Отримані дані,

потім перетворюються у візуальні цифрові зображення та можуть бути використані для загальних цілей, таких як управління водними ресурсами для використання під час поливу чи виявлення хвороб у рослин.

Variable rate technology (VRT) - технологія змінної швидкості - це компонент точного сільського господарства, який дає змогу використовувати дані безпосередньо. Це об'єднує сільськогосподарську техніку, системи керування та обладнання, щоб використовувати зростаючу кількість вхідною інформації в реальному часі та точному розташуванні. Точне сільське господарство з технологією VRT має водночас переваги економічні та для навколишнього середовища. Використання насіння, добрива, поживних речовин чи пестицидів тільки де і коли вони потрібні може значно зменшити витрати фермерів. Тим більш, негативний вплив на навколишнє середовище від перевикористання деяких хімічних речовин зменшується і використання деяких засобів потенційно може бути повністю усунено базуючись на аналізі даних. Постійна проблема, як використання нітрогену може бути вирішена, допомагаючи фермерам знайти правильну кількість мід недостатньою та завеликою. Використання системи виражається у замкнутому циклі, що складається з збору/аналізу даних, планування врожаю, виконання плану та аналізу результатів для наступного сезону.

ГІС має тисячі застосувань. Це постійно зростаюча область, яка навіть вже є глибоко інтегрованою в багато секторів підприємств. Вона продовжує робити інновації, які допомагають нашому повсякденному життю.

1.2 Змістовний опис і аналіз предметної області

Щільність грунту

Висока щільність грунту може бути серйозною формою погіршення його стану та може призвести до збільшення ерозії грунту та зниження врожаю. Ущільнення грунту зменшує розміри порового простору для повітря і води.

Близько 50% структури більшості грунтів складають тверді речовини (пісок, мул, глина та органічні речовини) і близько 50% - поровий простір.

Проблеми ущільненості ґрунту

Ущільнення грунту може погіршити зволоження грунту, дозрівання посівних, проникнення коренів та поглинання поживних речовин і води, що призводить до зниження врожаю. Ущільнення сільськогосподарських грунтів, спричинене людиною, може бути результатом використання обладнання при культивації ґрунту або результатом великої ваги сільськогосподарської техніки. Ущільнені грунти також можуть бути результатом природних процесів формування грунту.

Вплив ущільнення грунту

Використання сільськогосподарської техніки може призвести до ущільнення частинок грунту в меншому об'ємі. Оскільки частинки стискаються, простір між ними зменшується, тим самим зменшуючи простір, доступний в грунті для повітря і води.

Ущільнення грунту може мати ряд негативних ефектів на якість грунту та врожайність:

- зменшує пористі простори грунту;
- зменшує інфільтрацію води в грунт;
- знижує швидкість проникнення води в кореневі зони грунту;
- збільшує потенціал затримання води на поверхні грунту, виникнення поверхневих водойм, поверхневого заболочення ґрунтів і їх ерозії;
- знижує здатність ґрунту утримувати воду і повітря, які необхідні для росту і функціонування коренів рослин;
- перешкоджає зростанню коренів і обмежує об'єм ґрунту, що досліджується корінням;
- знижує потенціал урожайності.

У вологіші роки, ущільнення грунту може знизити провітрюваність грунту і призвести до збільшення втрат нітратного азоту шляхом денітрифікації, що полягає в перетворенні доступного в рослин нітрат азоту в газоподібну форму і втрати його в атмосфері. Цей процес відбувається, коли грунти перебувають в анаеробному стані і пори грунту заповнені переважно водою. Зменшення аерації грунту може вплинути на ріст і функціонування коренів і призвести до підвищеного ризику захворювання культур. Всі ці фактори призводять до збільшення стресу культур та втрати врожайності.

Вимірювання щільності

Грунтовий пенетрометр може бути використаний як діагностичний засіб для вимірювання ступеня і глибини ущільнення грунту. Зазвичай, тільки дослідники та консультанти з сільського господарства мають таке обладнання, а також знання та досвід для його правильного використання.

Пенетрометр зазвичай вводиться зі швидкістю 2.5 см в секунду. Важливо переконатись, що вимірювані ділянки мають подібну вологість до проби грунту. Якщо виявляються ущільнені ділянки, наступним кроком буде копання грунтових ям для дослідження грунту або використання гідравлічного блоку для обробки грунту для вивчення проблемних зон і підтвердження ущільнення грунту. Зміна текстури грунту, або сухий грунтовий шар можуть імітувати ущільнення при використанні пенетрометра.

Саме виміри зібрані пенетрометром на різних ділянках для кожної глибини будуть використовуватись даною ГІС для збереження та генерації статистичних звітів.

1.3 Аналіз успішних ІТ-проектів

Сьогодні існує багато різних ГІС для сільського господарства, що дозволяють зберігати та аналізувати просторові дані в різній мірі складності. Розглянемо деякі з них.

1.3.1 Sinergise

Це IT компанія, що фокусується на розробці великомасштабних ГІС рішень. Ядром бізнесу є підтримка урядів по всьому світу для ефективного управління їх сільськогосподарськими та земельними адміністративними процесами. Компанія надає ГІС інструменти, що можуть бути використаними як окремі продукти, або бути інтегрованими в більші проекти.

3D-переглядач Sinergise поєднує в собі аерофотозйомку з цифровою моделлю рельєфу або цифровою моделлю рельєфу для створення зручного інтерактивного 3D-перегляду ваших інтересів. Більше не обмежуючись повітряними картами, структура базового рельєфу може бути, нарешті, візуалізована дистанційно.

Оскільки технологія заснована на WebGL, простота і інтеграція браузера гарантовані - установка не потрібна. 3D-вигляд може додатково бути доповнений різними шарами даних, наприклад кадастровою інформацією. 3D-перегляд допомагає ідентифікувати ділянки і межі, які не чітко видно в 2D-зображеннях.

ТороCheck являє собою простий у використанні, потужний, крос-платформний, інструмент для перевірки просторових наборів даних, а також їх атрибути і метадані. Він є інструментом для використання адміністраторами даних, особливо в організаціях, які відповідають за створення, управління, розповсюдження та використання великих і важливих просторових наборів даних.

Продукт підтримує формати SHP та Oracle SDO. Також результати аналізу можуть бути експортованими в SHP файл.

Field Data Collection – це он-лайн система збору та управління даними - заснована на інфраструктурі Cloud. Система підтримуватиме централізоване управління даними та оцифровкою даних на основі аеро- та супутникових зображень.

Мобільна система збору польових даних - побудована на ОС Android, ця система буде підключатися до системи управління даними і підтримуватиме розподіл завдань (пакетів) серед геодезистів, керованого введення даних і безперебійної передачі даних. Процедури перевірки достовірності даних і процедури забезпечення якості обробки забезпечують виявлення та усунення помилок.

1.3.2 ESRI

ESRI є лідером серед GIS рішень. Компанія почала свою історію в 1969. І від того часу, інтегрувала ГІС рішення в різні сфери діяльності людини: розваги, реклама, географія, нерухомість, безпека та сільське господарство.

ArgGIS Tracker є частиною Esri Geospatial Cloud та є мобільним рішенням, що дозволяє організаціям отримувати дані про пересування персоналу на полі, відслідковувати де вони є та аналізувати де вони були. Шаблони треків дають людям, що приймають рішення, інформацію в реальному часу для підтримки важливої діяльності та здійснення аналізу.

ArgGIS Indoors - це повноцінна система для внутрішніх карт, що використовуються для з'єднаних робочих просторів. Вона забезпечує загальну картину для керівників, службового персоналу та інших працівників та відвідувачів, щоб розуміти, використовувати робоче середовище та ефективно ним керувати. Існують додатки для веб та мобільної платформ. ArgGIS Indoors допомагає створювати, кастомізувати, ділитися та використовувати карти робочого простору та дані про місцезнаходження, тож користувачі можуть керувати та створювати комфортне та привабливе середовище.

Ведення точного сільського господарства потребує детальної інформації про поле. Ключову інформацію отримують під тестування грунту та даних про врожайність для того, щоб визначити точну кількість поживних речовин та насіння для росту. Це допомагає збільшити врожайність та зменшити кількість відходів, тим самим збільшуючи рентабельність інвестицій. Використовуючи

просторову інформацію та відстежуючи результати щороку, фермер може постійно вдосконалювати та покращувати свої результати.

Рішення Esri допомагають збирати дані з нижче описаних етапів, аналізувати їх та будувати карти з просторовими даними з точними рекомендаціями.

Точне сільське господарство починається з регулярного тестування грунту. Фермер має розуміти кількість різних поживних речовин доступних в грунті, для того, щоб використовувати добрива тільки в необхідних кількостях. Результати зразків грунту інтерполюються таким чином, щоб вміст поживних речовин був зрозумілим для кожного місця в полі.

Детальна інформація отримується під час збору врожаю, включаючи врожайність в будь-якому певному місці. Ця інформація надає цінний внесок для розрахунку норм висіву та поправок на грунт на наступний рік, а також допомагає фермеру відслідковувати його результати. Досліджуючи врожайність протягом часу, фермер може розуміти які ділянки поля є більш родючими.

Деякі райони сільськогосподарських полів є історично більш продуктивними ніж інші, незалежно від типу культур та використаних добрив. Розуміючи, як продуктивність змінюється з часом та за місцем розташування, фермери можуть виділяти зони управління в межах своїх полів. Фермери можуть збільшити свою прибутковість, зосередивши свої інвестиції на найбільш продуктивних зонах управління.

Рекомендації щодо зміни складу грунту базуються на широкому спектрі факторів. До них відносять такі параметри, як існуючі поживні речовини, очікувана врожайність, тип культури попереднього року та попередня врожайність. Рекомендації часто змінюються залежно від зони управління. Оскільки ці дані про ці параметри доступні для кожного місця в межах поля, фермер може визначити необхідну кількість певного ресурсу, необхідного для кожного місця. Використовуючи тільки кількість, необхідну для кожного місця,

фермер не тільки збільшує врожайність та здоров'я культури, але й зменшує витрати.

Рекомендації базуються на деталізованих рівняннях і можуть бути доповнені додатковими факторами, такими як дані про кліматичну зону та тип грунту. Щоб ще більше збільшити врожайність, фермер може цілеспрямовано використовувати іншу від рекомендованої кількість ресурсу, щоб перевірити коригування норм застосування. Ці випробувальні ділянки повинні бути точно розташовані, щоб мати порівнювати врожайність в цих самих місцях.

1.4 Аналіз вимог до програмного забезпечення

Загалом система має мати такий функціонал:

- реєстрація користувачів;
- авторизація користувачів;
- збереження просторових даних про виміри в базі даних;
- збереження даних про користувачів в базі даних;
- збереження даних про пристрої в базі даних;
- збереження даних про поля в базі даних;
- збереження даних про компанії в базі даних;
- експорт даних у вигляді звіту;
- Загалом сервіс генерації звітів має мати такий функціонал;
- статистичний аналіз просторових даних про виміри щільності грунту зроблені певний користувачем пенетрометром на деякому полі протягом певного періоду;
- генерація звіту на основі вхідних та історичних просторових даних у форматі xml;
- генерація звіту на основі вхідних та історичних просторових даних у вигляді табличного документу;

– генерація звіту на основі вхідних та історичних просторових даних як документа у форматі pdf.

1.4.1 Розроблення функціональних вимог

Для опису процесу діяльності представимо діаграму діяльності (рисунок 1.1), що описує взаємодію користувача з системою для генерації репорті.

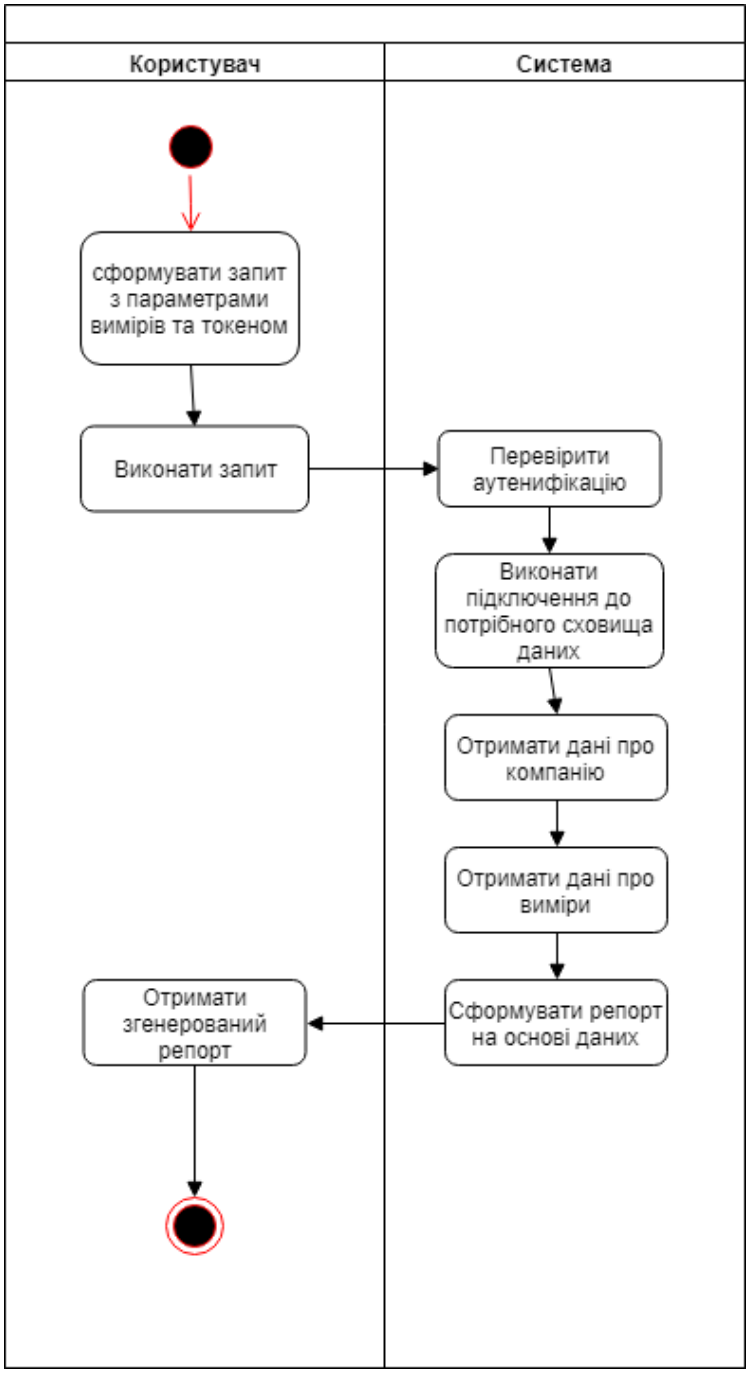


Рисунок 1.1 – Діаграма діяльності

Зручним і часто використовуваним способом задання вимог є діаграма прецендентів. На рисунку 1.2 наведення діаграму прецендентів для розроблюваного програмного забезпечення.

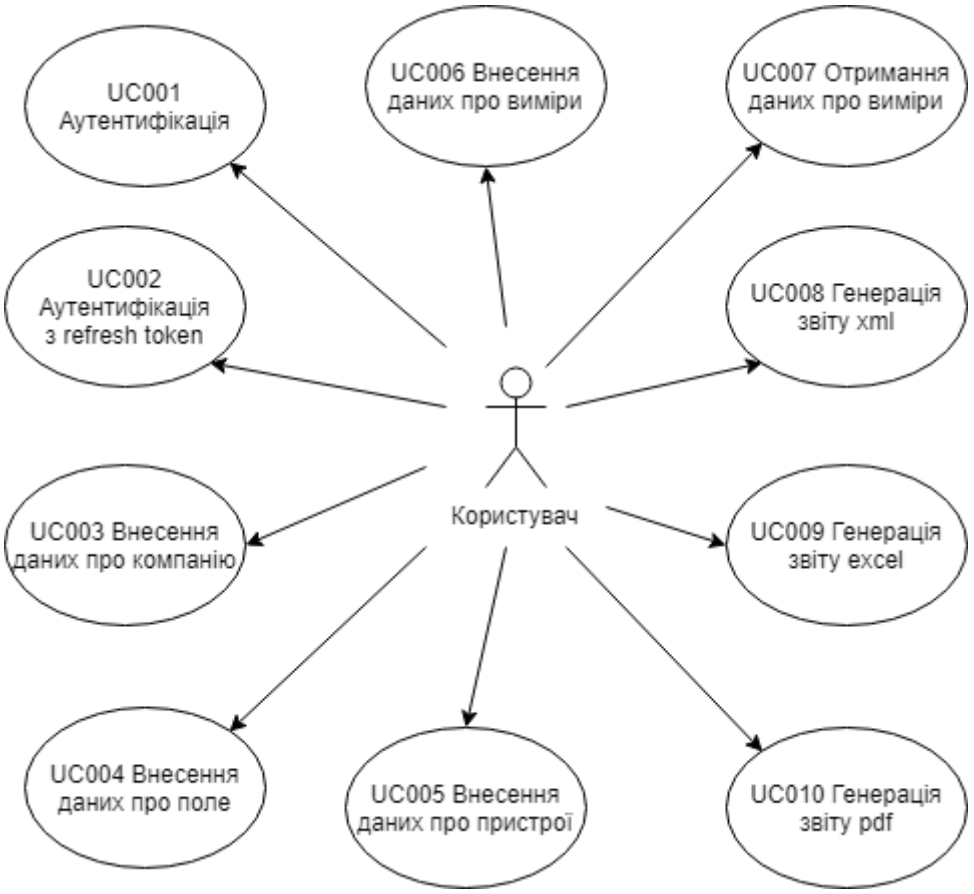


Рисунок 1.2 Діаграма прецендентів

В рішенні для геоінформаційної системи для щільності грунту передбачені наступні варіанти використання важливі для роботи з мікросервісом генерації звітів.

Таблиця 1.1 – Варіант використання UC001

Назва	Аутентифікація користувача з персональними даними
Опис	Користувач має можливість автентифікуватись з персональними даними для системного рішення для певної компанії з використанням протоколу OAuth2
Учасники	Користувач

Продовження таблиці 1.1

Передумови	Персональні дані про користувача зберігаються в захищеному сховищі компанії
Постумови	Користувач отримує пару токенів аутентифікації - access token і refresh token
Основний сценарій	Користувач надсилає запит на сервіс аутентифікації з ім'ям користувача і паролем Система надсилає відповідь з двома полями: access token і refresh token
Розширення сценаріїв	Сервіс виявляє, що користувача з вказаними даними не існує. Сервіс надсилає відповідь про помилку аутентифікації.

Таблиця 1.2 – Варіант використання UC002

Назва	Аутентифікація користувача з refresh token
Опис	Користувач має можливість автентифікуватись з refresh token, отриманим під час попереднього сеансу аутентифікації з використанням протоколу OAuth2
Учасники	Користувач, що має refresh token
Передумови	Персональні дані про користувача зберігаються в захищеному сховищі компанії
Постумови	Користувач отримує пару токенів аутентифікації - access token і refresh token

Продовження таблиці 1.2

Основний сценарій	Користувач надсилає запит на сервіс аутентифікації з refresh token Система надсилає відповідь з двома полями: access token і refresh token Refresh token може бути оновленим, якщо сервіс помітить, що скоро час його валідності закінчується
Розширення сценаріїв	Сервіс виявляє, що час активності refresh token закінчився. Сервіс надсилає повідомлення з помилкою про закінчення часу активності refresh token. В такому випадку користувач має здійснити аутентифікацію з використанням персональних даних. Сервіс виявляє, що отриманий refresh token не відповідає даним аутентифікації для жодного користувача. Сервіс надсилає повідомлення про помилку.

Таблиця 1.3 – Варіант використання UC003

Назва	Занесення даних в базу даних компаній
Опис	Автентифікований користувач має можливість додати інформацію про компанію в систему
Учасники	Автентифікований користувач
Передумови	Сервіс для доступу до сховища даних компаній розгорнутий і працює в корпоративному середовищі
Постумови	Інформація про компанію збережена в сховищі даних і доступна для користувачів сервісу

Продовження таблиці 1.3

Основний сценарій	Користувач надсилає дані про компанію на веб метод сервісу, що відповідальний за збереження даних
Розширення сценаріїв	Сервіс виявляє, що інформація про компанію неправильно сформовані. Сервіс надсилає відповідне повідомлення про помилку користувачу

Таблиця 1.4 – Варіант використання UC004

Назва	Занесення даних в базу даних про поле
Опис	Автентифікований користувач має можливість додати інформацію про поле в систему
Учасники	Автентифікований користувач
Передумови	Сервіс для доступу до сховища даних поле розгорнутий і працює в корпоративному середовищі. Інформація про компанію, що володіє полем збережена в системі та доступна користувачам
Постумови	Інформація про поле збережена в сховищі даних і доступна для користувачів сервісу
Основний сценарій	Користувач надсилає дані про поле на веб метод сервісу, що відповідальний за збереження даних про поля
Розширення сценаріїв	Сервіс виявляє, що інформація про поле неправильно сформовані. Сервіс надсилає відповідне повідомлення про помилку користувачу.

Таблиця 1.5 – Варіант використання UC005

Назва	Занесення даних в базу даних працівників
Опис	Автентифікований користувач має можливість додати інформацію про працівників компанії в систему
Учасники	Автентифікований користувач
Передумови	Сервіс для доступу до сховища даних працівників розгорнутий і працює в корпоративному середовищі. Інформація про компанію, де влаштовані працівники, збережена в системі та доступна користувачам
Постумови	Інформація про працівників збережена в сховищі даних і доступна для користувачів сервісу
Основний сценарій	Користувач надсилає дані про працівників на веб метод сервісу, що відповідальний за збереження даних
Розширення сценаріїв	Сервіс виявляє, що інформація про працівників неправильно сформовані. Сервіс надсилає відповідне повідомлення про помилку користувачу

Таблиця 1.6 – Варіант використання UC006

Назва	Занесення даних в базу даних пристроїв
Опис	Автентифікований користувач має можливість додати інформацію про пристрої (пенетрометри) в систему
Учасники	Автентифікований користувач

Продовження таблиці 1.6

Передумови	Сервіс для доступу до сховища даних пенетрометрів розгорнутий і працює в корпоративному середовищі. Інформація про компанію, якій належать пристрої, збережена в системі та доступна користувачам
Постумови	Інформація про пристрої збережена в сховищі даних і доступна для користувачів сервісу
Основний сценарій	Користувач надсилає дані про пристрої на веб метод сервісу, що відповідальний за збереження даних
Розширення сценаріїв	Сервіс виявляє, що інформація про пристрої неправильно сформовані. Сервіс надсилає відповідне повідомлення про помилку користувачу

Таблиця 1.7 – Варіант використання UC007

Назва	Занесення даних в базу даних вимірів
Опис	Автентифікований користувач має можливість додати інформацію про виміри щільності грунту в систему
Учасники	Автентифікований користувач
Передумови	Сервіс для доступу до сховища даних вимірів розгорнутий і працює в корпоративному середовищі. Інформація про поле, на якому проводилися виміри, збережена в системі та доступна користувачам. Інформація про пристрої, якими вимірялася щільність грунту, збережена в системі та доступна користувачам.

Продовження таблиці 1.7

Постумови	Інформація про виміри збережена в сховищі даних і доступна для користувачів сервісу
Основний сценарій	Користувач надсилає дані про виміри на веб метод сервісу, що відповідальний за збереження даних
Розширення сценаріїв	Сервіс виявляє, що інформація про виміри неправильно сформовані. Сервіс надсилає відповідне повідомлення про помилку користувачу

Таблиця 1.8 – Варіант використання UC008

Назва	Отримання даних про виміри
Опис	Автентифікований користувач має можливість зробити запит на отримання даних про виміри щільності
Учасники	Автентифікований користувач
Передумови	Сервіс для доступу до сховища даних вимірів розгорнутий і працює в корпоративному середовищі. Інформація про поле, на якому проводилися виміри, збережена в системі та доступна користувачам. Інформація про пристрої, якими вимірялася щільність грунту, збережена в системі та доступна користувачам. Інформація про працівників, які проводили виміри, збережена в системі та доступна користувачам
Постумови	Інформація про виміри збережена в сховищі даних і доступна для користувачів сервісу

Продовження таблиці 1.8

Основний сценарій	Користувач надсилає запит сервісу вимірів на отримання даних про виміри щільності на полі за певний періо

В мікросервісі генерації звітів передбачені наступні варіанти використання:

Таблиця 1.9 – Варіант використання UC009

Назва	Генерація звіту xml
Опис	Користувач, що пройшов етап автентифікації має можливість генерації звіту у форматі xml
Учасники	Автентифікований користувач
Передумови	Користувач автентифікований
Постумови	Користувач отримав звіт у форматі xml
Основний сценарій	Користувач надсилає запит з наступними параметрами: – ідентифікатор поля; – початкова дата вимірювань; – кінцева дата вимірювань; – одиниця вимірювання.
Розширення сценаріїв	Сервіс виявляє, що поля з вказаним ідентифікатором не існує. Сервіс не підтримує вказані одиниці вимірювання. Інтервал вимірювань невірний, або вимірювання не відбувались у вказаний період. Внутрішня помилка сервісу: не вдалося отримати відповідь від зовнішніх служб, помилка при форматуванні звіту. Система надсилає користувачу відповідь з відповідним повідомленням про помилку.

Таблиця 1.10 – Варіант використання UC010

Назва	Генерація звіту як табличного документу
Опис	Користувач, що пройшов етап автентифікації має можливість генерації звіту у форматі табличного документу
Учасники	Автентифікований користувач
Передумови	Користувач автентифікований
Постумови	Користувач отримав звіт у форматі табличного документу
Основний сценарій	Користувач надсилає запит з наступними параметрами: – ідентифікатор поля; – початкова дата вимірювань; – кінцева дата вимірювань; – одиниця вимірювання.
Розширення сценаріїв	Сервіс виявляє, що поля з вказаним ідентифікатором не існує. Сервіс не підтримує вказані одиниці вимірювання. Інтервал вимірювань невірний, або вимірювання не відбувались у вказаний період. Внутрішня помилка сервісу: не вдалося отримати відповідь від зовнішніх служб, помилка при форматуванні звіту. Система надсилає користувачу відповідь з відповідним повідомленням про помилку.

Таблиця 1.11 – Варіант використання UC011

Назва	Генерація звіту як pdf документа
Опис	Користувач, що пройшов етап автентифікації має можливість генерації звіту у форматі pdf документа
Учасники	Автентифікований користувач
Передумови	Користувач автентифікований
Постумови	Користувач отримав звіт у форматі pdf документа
Основний сценарій	Користувач надсилає запит з наступними параметрами: – ідентифікатор поля; – початкова дата вимірювань; – кінцева дата вимірювань; – одиниця вимірювання.
Розширення сценаріїв	Сервіс виявляє, що поля з вказаним ідентифікатором не існує. Сервіс не підтримує вказані одиниці вимірювання. Інтервал вимірювань невірний, або вимірювання не відбувались у вказаний період. Внутрішня помилка сервісу: не вдалося отримати відповідь від зовнішніх служб, помилка при форматуванні звіту. Система надсилає користувачу відповідь з відповідним повідомленням про помилку.

Таблиця 1.12 – Опис функціональної вимоги REQ001

Номер	REQ001

Продовження таблиці 1.12

Назва	Аутентифікація користувачів
Опис	Для роботи з системою, користувач має пройти аутентифікацію

Таблиця 1.13 – Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Можливість збереження вимірів
Опис	Користувач має змогу надіслати дані про виміри сервісу вимірів для подальшого збереження і використання

Таблиця 1.14 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Можливість збереження даних про компанії
Опис	Користувач має змогу надіслати дані сервісу компаній для подальшого збереження і використання

Таблиця 1.15 – Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Можливість збереження даних про поля
Опис	Користувач має змогу надіслати дані сервісу полів для подальшого збереження і використання

Таблиця 1.16 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Можливість збереження пристроїв
Опис	Користувач має змогу надіслати дані сервісу пристроїв для подальшого збереження і використання

Таблиця 1.17 – Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Можливість специфічного запиту на отримання вимірів
Опис	Користувач має змогу надіслати запит з параметрами для отримання вимірів за специфічний період, конкретного поля та певним пристроєм

Таблиця 1.18 – Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Можливість генерації xml репорту
Опис	Користувач має змогу надіслати запит сервісу генерації репортів для генерації репорту у форматі xml

Таблиця 1.19 – Опис функціональної вимоги REQ008

Номер	REQ008
Назва	Можливість генерації excel репорту
Опис	Користувач надіслає запит сервісу генерації excel репорту

Таблиця 1.20 – Опис функціональної вимоги REQ009

Номер	REQ009
Назва	Можливість генерації pdf репорту
Опис	Користувач має змогу надіслати запит сервісу генерації репортів для генерації репорту у форматі pdf

На рисунку 1.1 зображено залежність між функціональними вимогами

Рисунок 1.3 – Схема структурна функціональних вимог

На рисунку 1.4 зображено залежність між функціональними вимогами та сценаріями використання.

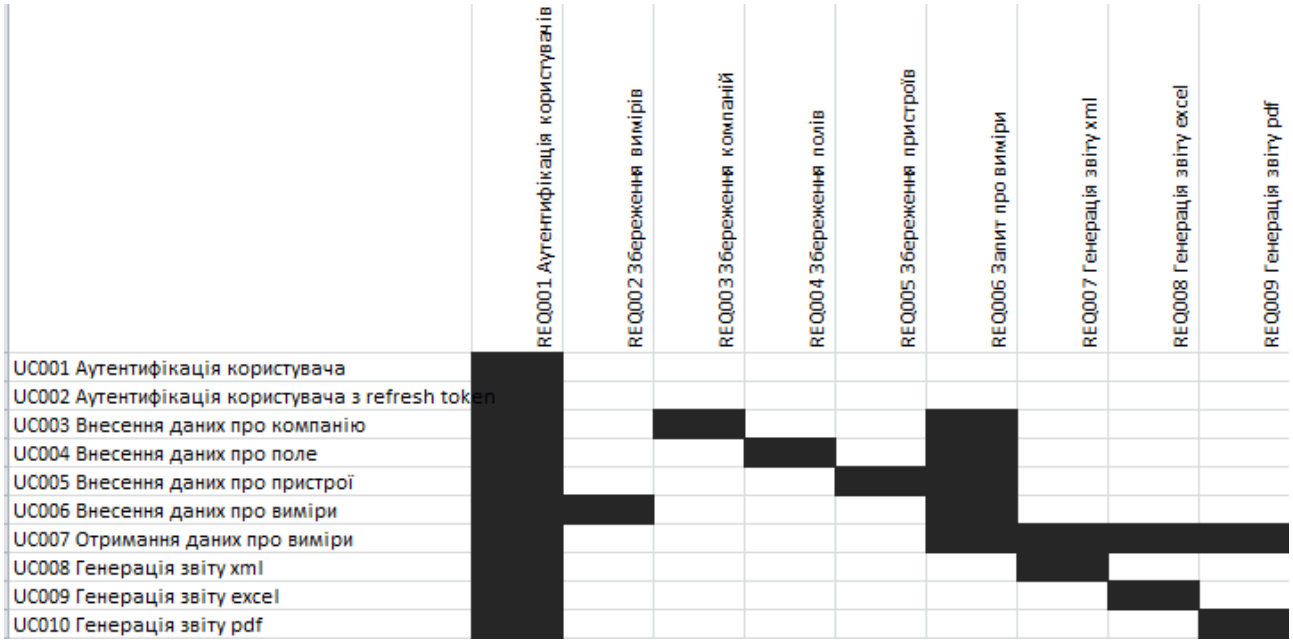


Рисунок 1.4 – Матриця трасування

1.4.2 Розроблення нефункціональних вимог

Сервіс для генерації звітів має відповідати наступним нефункціональним вимогам:

– локалізація звітів - генерація звітів українською мовою з можливістю розширення списку підтримуваних мов;

– передача повідомлень між сервісами використовуючи підхід REST;

– передача даних між сервісами по захищеному каналу використовуючи протокол транспортного рівня TLS;

– передача повідомлень з клієнтом по захищеному каналу використовуючи протокол транспортного рівня TLS;

– передача текстових даних між сервісами використовуючи формат JSON.

1.4.3 Постановка комплексу завдань модулю

Розроблюване програмне забезпечення призначене для фермерів та сільськогосподарських підприємств, що прагнуть автоматизувати процесу керування, обробки та представлення просторових даних про щільність грунту

на різній глибині і одночасно забезпечити високий рівень захисту даних від неконтрольованого та несанкціонованого доступу.

Метою розробки є побудова мікросервісів геоінформаційної хмарної системи для генерації репортів та керування доступом до вимірів про щільність ґрунту. Для цього група компонетів має виконувати наступні задачі:

- додавання просторових даних про виміри пенетрометром щільності грунту на різній глибині;
- збереження даних для швидкого подальшого доступу;
- обмеження доступу до інформації певної компанії за отриманим токеном;
- генерація звіту з статистичними та графічними додатками для вимірів щільності грунту за певний період на заданому полі у форматах xml, excel, pdf.

1.5 Висновки по розділу

Щільність грунту безпосередньо має вплив на ці фактори, а отже контроль на підтримка щільності у допустимих межах має входити в комплекс робіт підтримки здоров'я грунту. Інструменти такі як пенетрометри допомагають полегшити процес вимірювання щільності грунту на різній глибині. В свою чергу, геоінформаційні системи повинні ефективно та зберігати історичні просторові дані та по можливості автоматизовувати часомістку та кропітливу роботу, щоб фермер міг концертувати зусилля на прийманні рішень та розвитку сільськогосподарського підприємств.

Розробка гнучкої системи, що не потребує складної інфраструктури та висококваліфікованого персоналу для обслуговування зменшила б поріг входу фермерів для адаптації інформаційних рішень та загалом покращило б якість використання сільськогосподарських угідь.

2 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Процес отримання вимірів сервісом (рисунок 2.1):

– сервіс надсилає запит на сервіс вимірів з токеном;

– сервіс вимірів надсилає запит на сервіс вимірів для перевірки валідності токену;

– після цього сервіс вимірів надсилає токен на сервіс database per tenant для отримання посилання на сховище даних, до якої треба підєднатися;

– сервіс вимірів робить запит до сховища даних;

– сервіс вимірів перетворює дані зі сховища даних відповідно до потреб доменної моделі;

– сервіс вимірів повертає результат з вимірами.

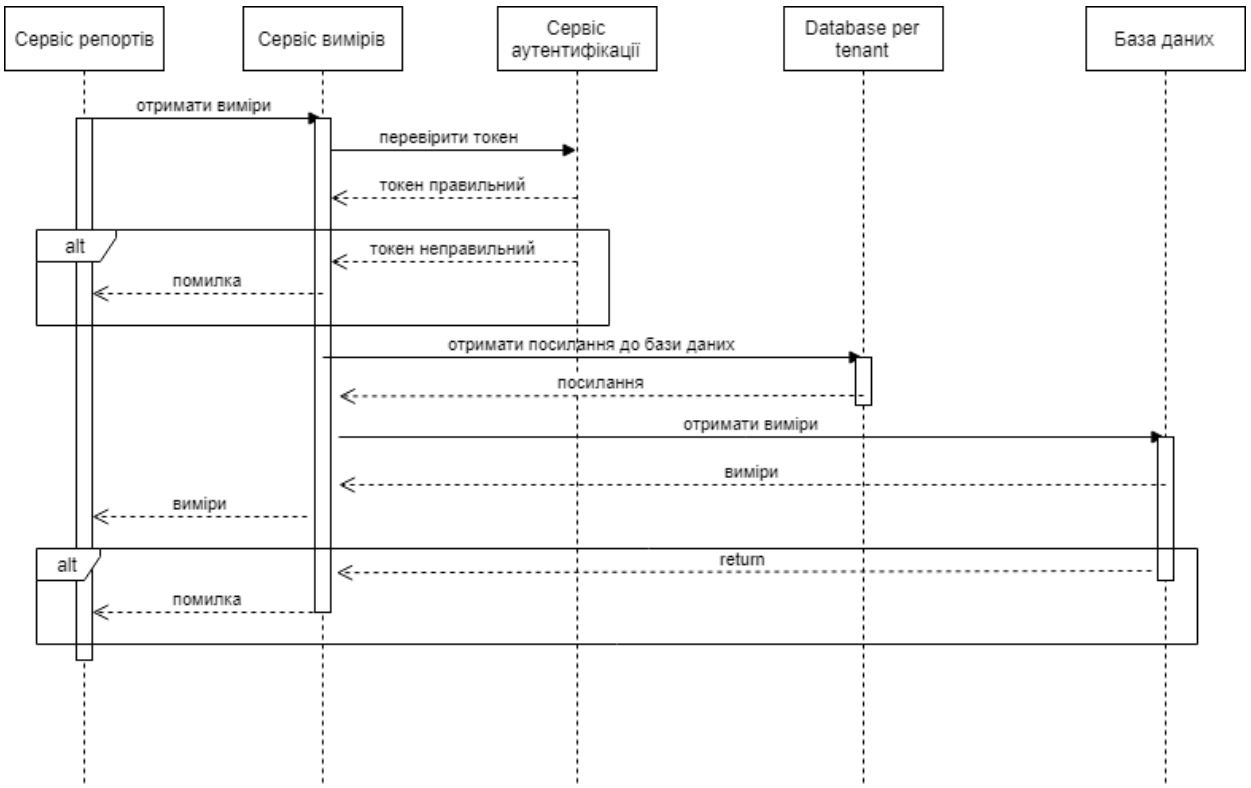


Рисунок 2.1 – Процес отримання вимірів

Для користування сервісом, користувач має успішно пройти аутентифікацію (рисунок 2.2).

– користувач має access token для тимчасової аутентифікації;

– якщо користувач не має access token, або цей токен вже не активний, користувач має отримати новий;

– якщо користувач має активний refresh token, він може отримати новий access token;

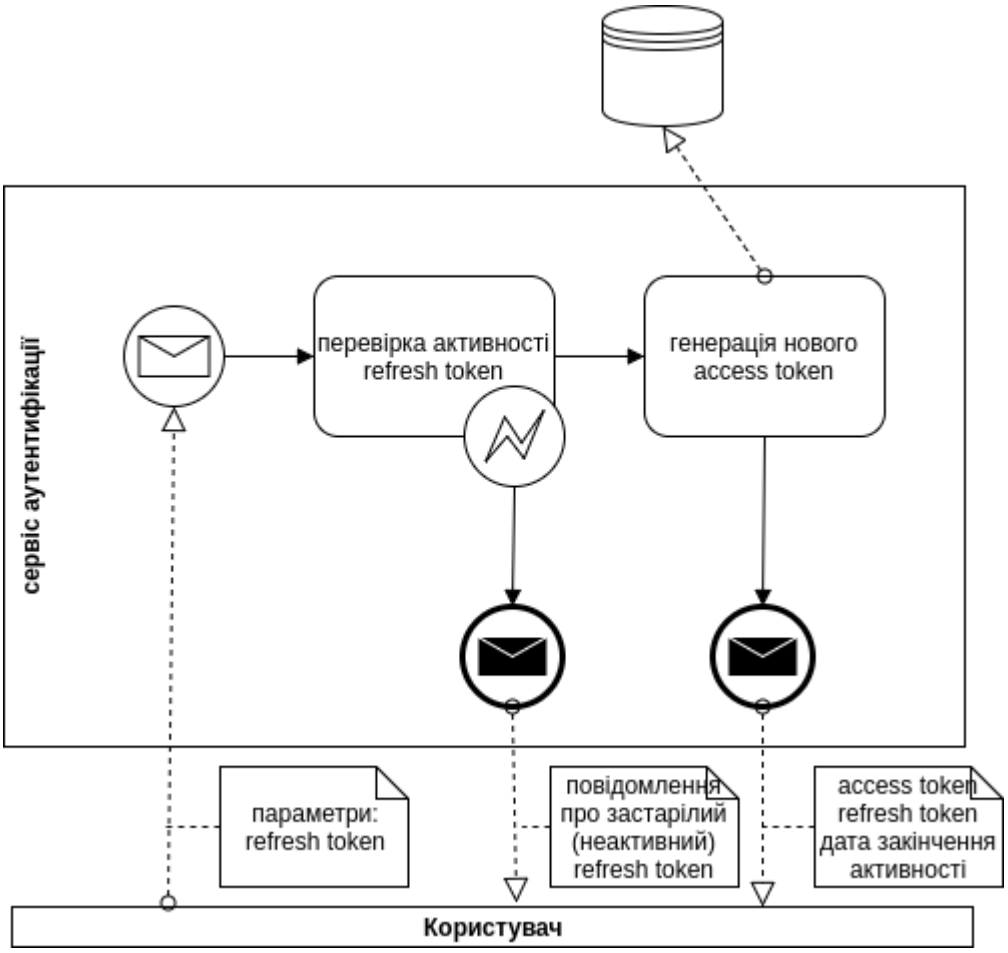


Рисунок 2.2 – Схема оновлення токену

– інакше користувач має отримати пару access token і refresh token використовуючи персональні дані (рисунок 2.3): ім'я користувача та пароль;

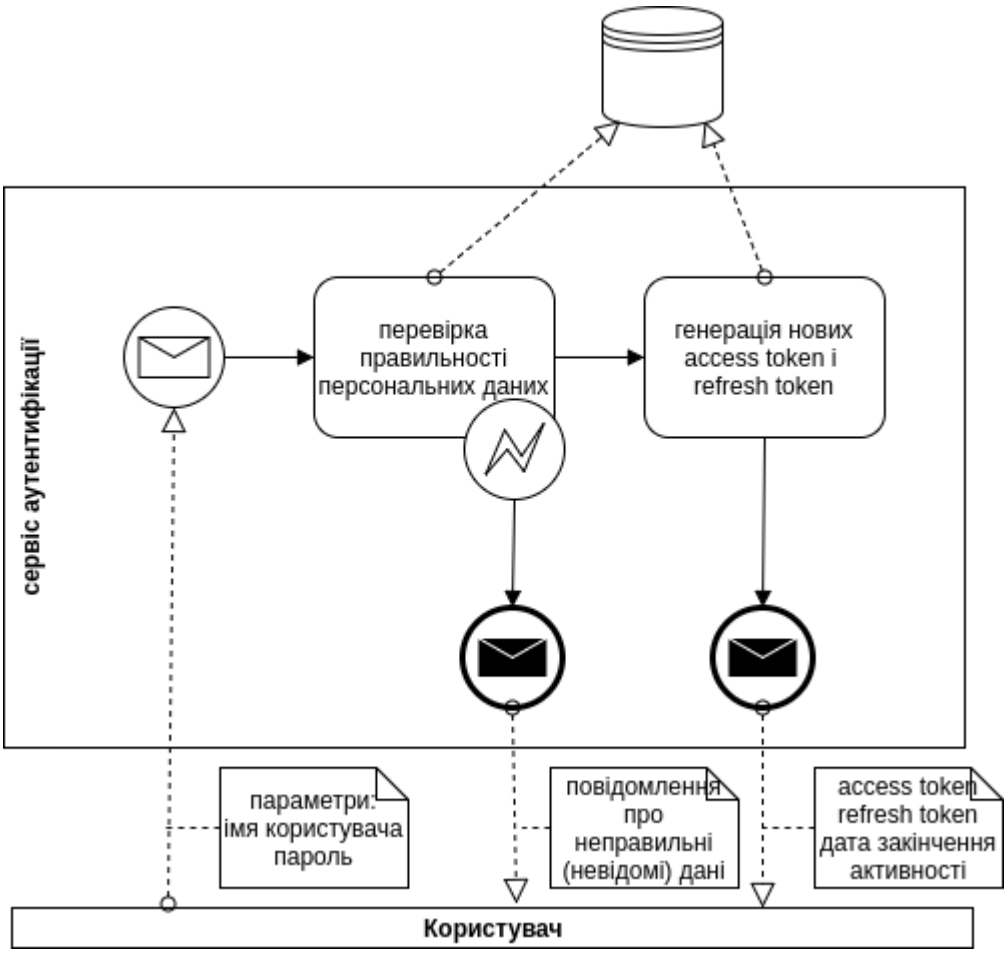


Рисунок 2.3 – Схема генерації токену

– отриманий access token користувач надсилає в заголовку веб запиту.

Після успішного проходження аутентифікації, користувач може робити запити на генерацію звіту

Рисунок 2.4 – Схема послідовності алгоритму генерації звіту

На початку процесу генерації репорту (рисунок 2.4) користувач надсилає запит на відповідний веб метод сервісу з наступними параметрами:

- початкова дата вимірів;
- кінцева дата вимірів;
- ідентифікатор поля;

- одиниці вимірювання;
- access token в заголовку.

Потім сервіс перевіряє правильність вхідних параметрів. Якщо запит сформований неправильно, сервіс надсилає користувачу помилку з відповідним повідомленням. Сервіс робить запит на отримання даних про компанію. Далі сервіс робить запит на отримання даних про поля. Якщо дані про поля недоступні, сервіс надсилає користувачу відповідне повідомлення про помилку. Якщо дані про поля недоступні, сервіс надсилає користувачу відповідне повідомлення про помилку. Сервіс перетворює дані, відповідно до вимог формату звіту. Також обраховується статистична аналітика. Сервіс формує звіт і надсилає його у відповіді користувачу.

2.2 Архітектура програмного забезпечення

2.2.1 Підхід до архітектури системи

Як один з варіантів сервісно-орієнтованої архітектури (SOA), мікросервіси це це архітектурний стиль в якому програми докомпозовані в слабо зв'язані сервіси. З поділом на сервіси з обмеженими та чіткими задачами та легковісними протоколами, мікросервіси пропонують підвищену модульність, що полегшує розробку, тестування, розгортання і, що більш важливо, зміну і підтримку програм.

Клієнт-сервісна модель була чудовим вибором, коли програмні рішення для персональних комп'ютерів складали основну частину забезпечення. Але з ростом популярності мобільних пристроїв та хмарних технологій, бекенд дані має завжди бути доступними широкому набору пристроїв.

З монолітною архітектурою, всякий час, коли відбувається зміна, потрібно оновити всю програму. Гірше того, так як все привязане, до єдиної кодової бази, неможливо масштабувати окрему функцію, або компонент. Вся програма повинна бути масштабована, що призводить до значних втрат продуктивності.

З мікросервісною архітектурою, код поділений на незалежні сервіси, що працюють в різних процесах. Вихідні дані з одного сервісу використовуються як вхідні дані для іншого в оркестрації незалежних сервісів.
Навіть, для організацій, що не є такими великими, як Netflix чи Amazon, мікросервіси приносять велику користь.

2.2.1.1 Підвищена стійкість

З мікросервісною архітектурою, вся програма є децентралізованою на відокремлені сервіси, що діють, як окремі суб'єкти. На відміну від монолітної архітектури, де помилка в коді впливає на роботу більш ніж одного компоненту чи функції, існує мінімальний вплив від помилки в мікросервісі. Навіть, якщо декілька служб знаходяться на технічному обслуговуванні, це не впливає на користувачів.

2.2.1.2 Покращена масштабованість

Масштабування є ключовим аспектом для мікросервісів. Так як кожен сервіс є окремим компонентом, він може бути масштабованим без необхідності масштабування всієї програми. Критичні для функціонування бізнесу сервіси можуть бути розгорнутими на декількох серверах для збільшеної доступності та ефективності і не впливати на продуктивність інших сервісів.

2.2.1.3 Менший час запуску продуктів на ринок

Так як мікросервіси працюють зі слабо зв'язаними сервісами, розробка нової функціональності потребує змін тільки в певних сервісах. Розробляючи програму малими інкрементами, що незалежно тестуються та розгортаються, час виходу продукту на ринок зменшується.

2.2.1.4 Безперервна доставка

На відміну від монолітних програм, в яких відповідальні команди працюють над розробкою конкретних аспектів - користувацьким інтерфейсом,

базою даних, серверною логікою - мікросервісний підхід використовує крос-функціональні команди, що працюють над повним циклом розробки використовуючи модель постійної доставки. Коли команда розробки і тестування працюють постійно над окремим сервісом, тестування і перевірка якості стають простими і швидкими. З підходом інкрементної розробки код постійно доставляється в робоче середовище будучи протестованим і розгорнутим.

2.2.1.5 Опис системи

Система містить сервіс аутентифікації, сервіс полів і вимірів, сервіс компаній та сервіс генерації звітів. На рисунку 2.5 зображено залежності між елементами системи

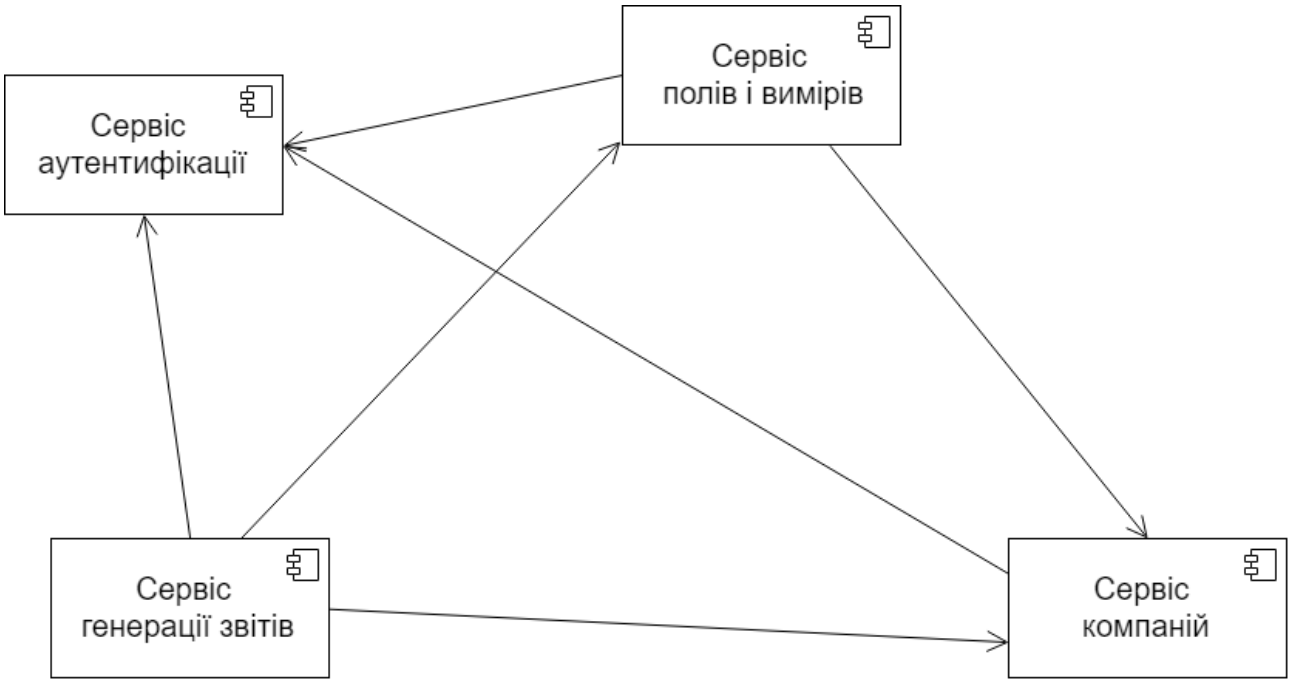


Рисунок 2.5 – Схема залежностей сервісів системи

2.2.2 Підхід до архітектури сервісу

Протягом останніх декількох років, сформувалося декілька головних ідей щодо архітектури компонентів. Вони включають:

- Гексагональна архітектура;
- Шарова архітектура;
- Кричуща архітектура;
- DCI;
- BCE.

Попри те, що ці архітектури відрізняються в деталях, вони мають багато спільного. Вони всі мають спільну ціль - розподіл обов'язків. Всі вони досягають цього шляхом поділу програмного забезпечення на шари. Кожен має принаймні один шар бізнес правил та інший для зовнішніх інтерфейсів. Кожна з цих архітектур формують сервіс, що є:

- незалежним від фреймворків. Архітектура не залежить від існування певної бібліотеки чи багатофункціонального програмного забезпечення. Це дозволяє використовувати фреймворки як інструменти і не обмежувати сервіс їх рамками;
- легким для тестування. Бізнес правила можуть бути протестованими без графічного інтерфейсу, бази даних, веб серверу чи інших зовнішніх елементів;
- незалежним від користувацького інтерфейсу. Користувацький інтерфейс може бути легко зміненим, не впливаючи на інші частини системи. Наприклад, графічний інтерфейс може бути заміненим на інтерфейс командного рядка без впливу на бізнес правила;
- незалежним від бази даних. Нові абстракції створюються для сховищ даних, що дозволяє бізнес правилам не залежати від конкретної імплементації (Oracle, Mongo, CouchDB, BigTable та інші);
- незалежним від зовнішніх сервісів. Бізнес правила просто не знають про існування будь-чого з зовнішнього світу;

Система поділяється на 4 шари (рисунок 2.6):

- шар корпоративних бізнес правил. Правила, що стосуються всього бізнесу. Тобто всіх IT рішень та внутрішніх процесів компанії. Це ті правила,

що визначають роботу всього бізнесу, а не окремих інформаційних систем. Тільки зміна роботи бізнесу впливає на зміни на даному шарі;

– шар бізнес правил системи. Правила, які поширюються на конкретну систему. Зміни в операціях системи впливають на зміни на даному шарі;

– шар адаптерів. Шар, що надає абстракції, які перетворюють дані з зовнішнього шару в дані, зручні для користування на рівнях бізнес правил. Тобто шар відділяє сутності конкретних імплементацій від потрапляння на рівень бізнес правил;

– шар фреймворків та служб. На зовнішньому рівні представлені класи, що взаємодіють з зовнішніми службами, або фреймворками. Це місце де класи, що працюють з конкретними базами даних, веб серверами, користувацьким інтерфейсом.

Також важливо, що зовнішні шари залежать від внутрішніх, але не навпаки. Це досягається переважно використанням абстракцій, таких як інтерфейси та застосування принципів IoC та DI.

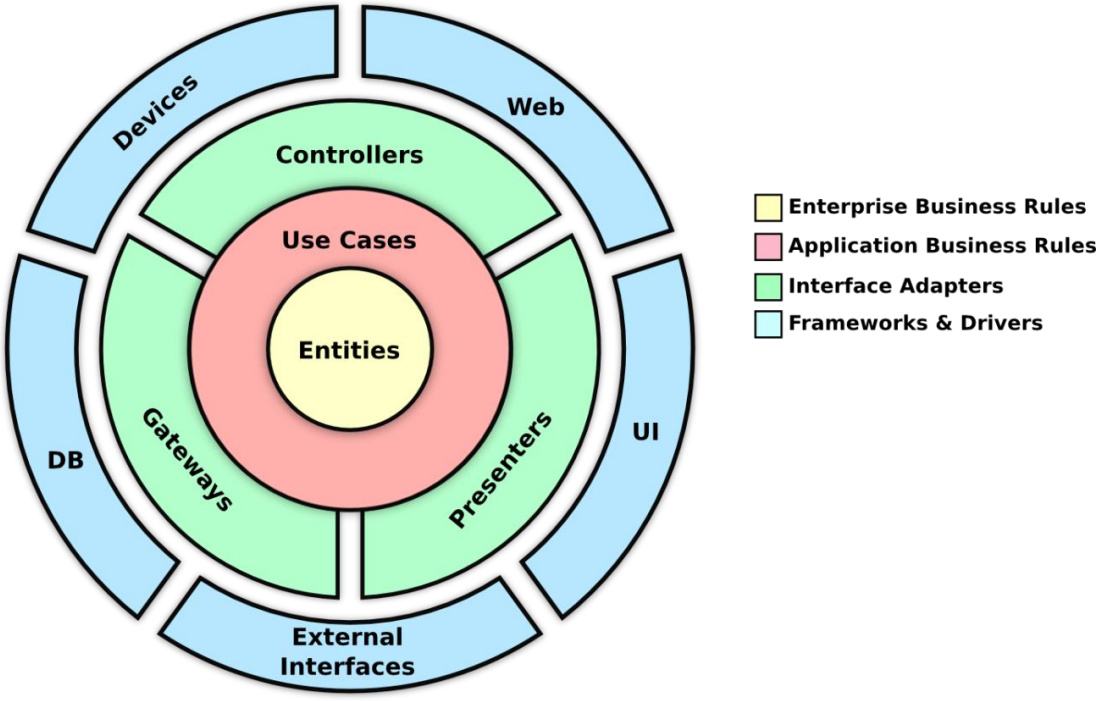


Рисунок 2.6 – Шари чистої архітектури

Мікросервіс звітів спроектований базуючись на принципах Чистої архітектури. Це можна чітко побачити на діаграмі пакетів та залежностей між ними.

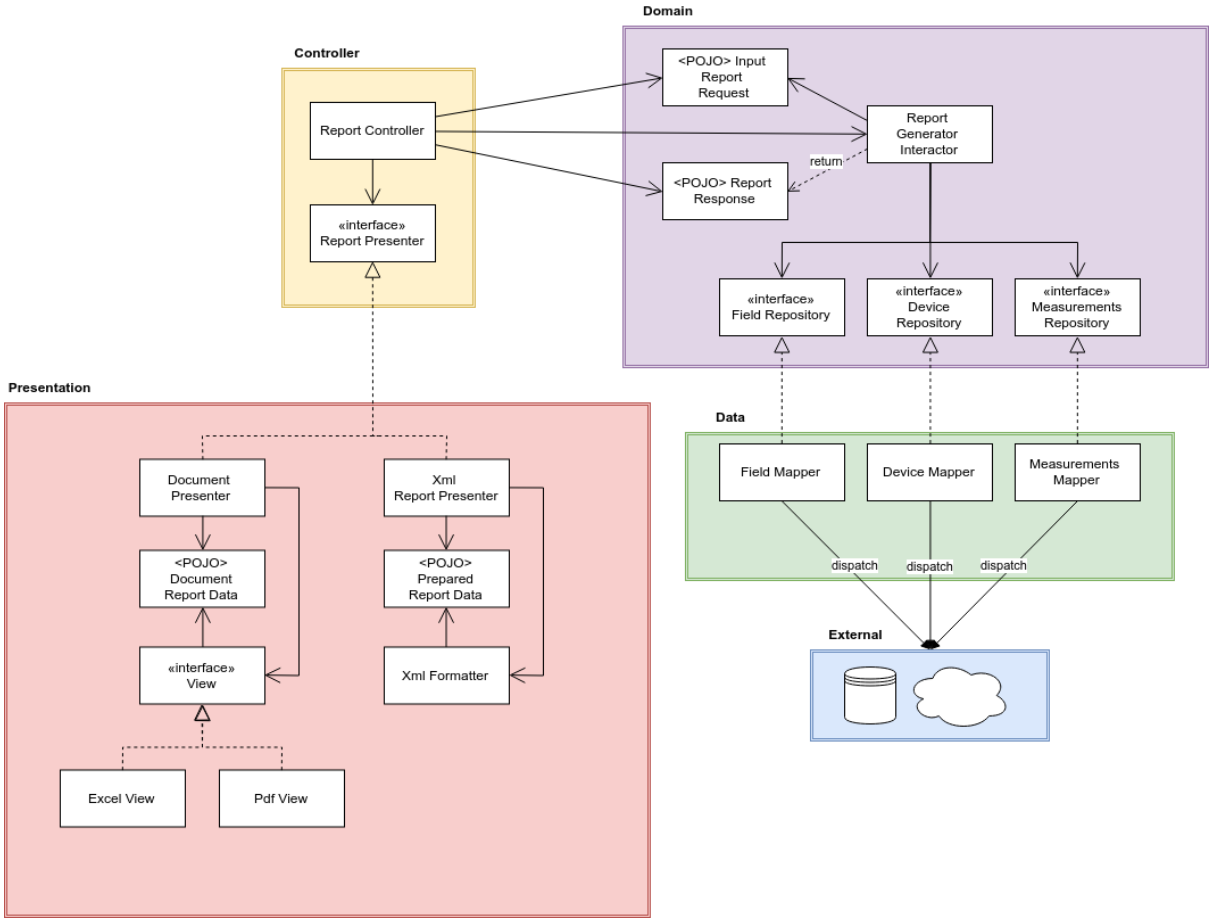


Рисунок 2.7 – Схема шарів сервісу генерації звітів

Пакет domain (рисунок 2.7) містить класи, що представляють бізнес правила. Також пакет містить класи, що представляють дані про звіт. Ці класи є простими POJO, що не мають залежностей на зовнішні бібліотеки. Пакет містить інтерфейси для сховищ даних. Таким чином пакет не має залежностей на інші пакети.

Пакет controller містить класи, що представляють інтерфейс сервісу. В них сконцентрована поведінка та отриманню веб запитів. Класи мають залежності на пакет domain. Так як після валідації вхідних даних відправляють їх на обробку класам бізнес правил. При цьому класи пакету controller залежать тільки від інтерфейсів пакету domain, слідуючи принципу IoC. Також пакет

controller не має залежності на пакет з реалізацією сховищ даних. Controller агностичний до форми бізнес правил і тільки слідує їх вимогам.

Пакет controllers оголошує інтерфейси для презентерів - спеціальних класів, що перетворюють дані отримані з шару бізнес правил у дані зручні для конкретного представлення, так як бізнес правила не знають про варіанти представлення даних користувачам. Презентери не містять жодної логіки, лише форматування та перетворення даних та відображення їх у представленні. Презентери також не знають про вигляд представлень, так як взаємодіють з інтерфейсом View.

Імплементації View займаються створенням документів, стилів та розміщенню даних, отриманих від презентерів. Після цього представлення отримують користувачі сервісу.

Пакет data містить класи, що імплементують інтерфейси сховищ даних, оголошених в пакеті domain. Ці класи безпосередньо звертаються до сховищ даних, таких як бази даних, файли у локальній чи віддаленій файловій системі, здійснюють веб запити зовнішнім сервісам, чи зберігають дані в пам'яті. Ці деталі обмежені пакетом data і ніяким чином не потрапляють на внутрішні рівні. Також класи data не реалізують бізнес правила, а лише вміють надати потрібні дані на вимогу.

2.2.3 Опис класів, інтерфейсів та методів сервісу

Проект реалізує принципи Чистої архітектури, в тому числі поділ на рівні. Детальний опис, класів/інтерфейсів кожного рівня наведено в таблицях 2.1- 2.5

Таблиця 2.1 – Інтерфейси сервісу генерації звітів

Інтерфейс	Опис
ReportPresenter	Інтерфейс, який є презентером для формування звіту різних форматів

Продовження таблиці 2.1

CompanyRepository	Інтерфейс рівню надання даних для отримання даних про компанії, що ізолює класи бізнес-логіки від способу доступу до даних
DeviceRepository	Інтерфейс рівню надання даних для отримання даних про пристрої, що ізолює класи бізнес-логіки від способу доступу до даних
FieldRepository	Інтерфейс рівню надання даних для отримання даних про поля, що ізолює класи бізнес-логіки від способу доступу до даних
MeasurementRepository	Інтерфейс рівню надання даних для отримання даних про виміри, що реалізує паттерн проектування Репозиторій та ізолює класи бізнес-логіки від способу доступу до даних
UserRepository	Інтерфейс рівню надання даних для отримання даних про користувачів, що ізолює класи бізнес-логіки від способу доступу до даних
View	Інтерфейс передачі та відображення звітів у певному форматі

Таблиця 2.2 – Класи моделей шару domain

Клас	Опис
Company	Анемічна модель, що відображає сукупні дані про компанію, які є важливими в рамках даної доменної області

Продовження таблиці 2.2

Клас	Опис
Field	Анемічна модель, що відображає сукупні дані про поле, які є важливими в рамках даної доменної області
FieldGroup	Анемічна модель, що відображає сукупні дані про групу полів, які є важливими в рамках даної доменної області
Location	Анемічна модель, що відображає сукупні дані про місцезнаходження (довгота і широта), які є важливими в рамках даної доменної області
Measurement	Анемічна модель, що відображає сукупні дані про вимір щільності ґрунту на полі за допомогою пристрою, які є важливими в рамках даної доменної області
User	Анемічна модель, що відображає сукупні дані про користувача пристрою, які є важливими в рамках даної доменної області
PreparedReportData	Клас, які містить дані про виміри, зроблені на групі полів за певний період, зібрані з інших сервісів та сховищ даних, які будуть відображені у звіті
FatDevice	Клас, що представляє розширену модель даних про пристрій та містить інформацію про виміри зроблені пристроєм, а також користувачів, які виконували вимірювання
Unit	Анемічна модель одиниці вимірювання щільності ґрунту

Продовження таблиці 2.2

Клас	Опис
ChartData	Клас, що містить перетворені підготовлені дані про виміри за певний період, для відображення на графіку залежності щільності грунту від глибини виміру
StatisticDensities	Клас, що містить аналітичні дані про мінімальні, максимальні та середні значення про щільність грунту для кожної глибини вимірювання
MultiChartData	Клас, що містить перетворені підготовлені дані про виміри за певний період, для відображення на графіку декількох аналітичних залежностей щільності грунту від глибини виміру
Series	Клас, що містить набір значень по вертикальній та її назву для графіку залежності щільності грунту від глибини виміру

Таблиця 2.3 – Класи шару domain

Клас	Опис
PrepareReportDataInteractor	Клас, що містить логіку по агрегації інформації необхідної для репорту з різних джерел даних. Також займається початковою фільтрацією та форматуванням даних відповідно до бізнес-вимог.
InputReportRequestMapper	Клас, що містить поведінку по перетворенню вхідних даних в модель, яка використовується на рівні доменної логіки

Продовження таблиці 2.3

InvalidRequestException	Виключення, що виникає в мікросервісі, при неправильному формату вхідних даних
GlobalExceptionHandler	Клас, що займається глобальним відловленням виключень мікросервісу та відновленням нормальної поведінки або сповіщенням клієнта про помилковий стан системи

Таблиця 2.4 – Класи контролеру шару view

Клас	Опис
ReportController	Клас, що містить методи, які є вхідними точками REST API мікросервісу
ApplicationException	Базове виключення, що виникає при неочікуваній або небажаній події або стану під час роботи сервісу

Таблиця 2.5 – Класи шару view

Клас	Опис
InputReportRequest	Клас, що зберігає перевірені та відформатовані вхідні дані про конфігурацію звіту.
CellStyleProvider	Клас, що налаштовує та зберігає інформацію про стилі, що можуть бути застосовані до різних елементів табличного документу.

Продовження таблиці 2.5

Клас	Опис
ChartCreator	Клас, що займається створенням, розташуванням, конфігурацією та заповненням графіку для однієї або більше серій в табличному документі.
ExcelHelper	Клас, що містить допоміжні методи для створення та конфігурації елементів табличного документу.
ExcelView	Клас, що містить представлення репорту у вигляді табличного документу, реалізує інтерфейс View
DocumentReportDataMapper	Клас, що перетворює початково зібрані дані в формат, зручний для використання при формуванні складних звітів
MathUtils	Клас, що містить методи для математичних перетворень
StringFormatUtils	Клас, що містить методи для форматування текстових даних
UrlFormatter	Клас, що займається формуванням веб-посилань
ChartCreator	Клас, що займається створенням, розташуванням, конфігурацією та заповненням графіку для однієї або більше серій в pdf документі.
PdfHelper	Клас, що містить допоміжні методи для створення та конфігурації елементів pdf документу.

Продовження таблиці 2.5

Клас	Опис
PdfView	Клас, що містить представлення репорту у вигляді pdf документу, реалізує інтерфейс View
DocumentPresenter	Клас, що містить логіку по перетворенню початкових даних для звіту та формування складного звіту у певному форматі
Labels	Клас, що містить текстові дані, що використовуються у складних звітах для пояснення його елементів
View	Інтерфейс, який є абстракцією для різних форматів представлення звітів.
XmlFormatter	Клас, що займається формування звіту xml з початкових даних за узгодженою схемою документу
XmlReportPresenter	Клас, що містить логіку по перетворенню початкових даних для звіту та формування звіту у форматі xml
DateParser	Клас, що містить методи по перетворення дати і часу у різні формати
Application	Головний клас, що містить метод запуску мікросервісу та додаткову конфігурацію

2.3 Конструювання програмного забезпечення

2.3.1 Оркестрація контейнерів

Контейнери значно збільшили гнучкість роботи хмарних застосунків як на фізичних так і на віртуальних середовищах. Контейнери упаковують сервіси, що є складають програму, і роблять їх портативними для різних обчислювальних середовищ, як для тестового так і для промислового використання. З контейнерами легко і швидко адаптувати сервіс під вимоги до поточного навантаження. І оскільки контейнери безпосередньо використовують ресурси хостової ОС, вони значно легші, ніж віртуальні машини. Це означає, що контейнери більш ефективно використовують серверну інфраструктуру.

Але хоча API контейнерів добре спроектоване для управління окремим контейнером, складність роботи значно збільшується коли коли йде мова про управління застосунками, що можуть складатися з сотень контейнерів, що розташовані на різних машинах. Контейнери мають керуватися та об'єднуватися ззовні для таких задач як планування, балансування навантаження і поширення, і для таких задач такі інструменти для оркестрації контейнерів як Kubernetes мають місце.

Відкрите програмне забезпечення для розгортання, масштабування та управління контейнеризованими програмами, Kubernetes об'єднує розробку програмного забезпечення та його обслуговування в рамках серверної інфраструктури разом. Використовуючи декларативні, незалежні від інфраструктури команди для того, щоб описувати як програми компонуються, як вони взаємодіють і як вони керуються, Kubernetes дає змогу на порядок збільшити ефективність сучасних програмних систем.

Kubernetes був створений компанією Google базуючись на власному досвіді з використання контейнерів в робочому середовищі. Продукт був переданий громаді як відкрите програмне забезпечення в 2014 і від того розвивається під

наглядом організації CBCF. Kubernetes є стандартом оркестрації контейнеризованих додатків більшості PaaS, таких як GCP, AWS і Azure.

Традиційно, програми були тісно зв'язаними з інфраструктурою, що спричиняло фактичну неможливість легкої зміни інфраструктури, попри потенційні переваги. Kubernetes усуває жорстку прив'язаність до конкретного власника інфраструктури, надаючи можливість для контейнеризації без обмежень.

Контейнери дозволяють програмам бути поділеними на менші частини досяючи кращого розмежування проблем. Такий модульний підхід пришвидшує розробку командами, сфокусованими та відповідальними за конкретний контейнер. Це дозволяє також ізолювати залежності і ширше використовувати добре налаштовані, менші компоненти. Але це не може бути досягнуто виключно контейнерами; це вимагає системи для інтеграції та організації цих модульних частин. Kubernetes досягає цього частково завдяки концепту Pod - колекції контейнерів, які контролюються як єдина програма.

Контейнери використовують спільні ресурси, такі як файлова система, простори імен ядра і IP адреси. Завдяки тому, що контейнери можуть бути розміщені таким чином, Kubernetes усуває бажання поміщати забагато різної функціональності в єдиний сервіс.

Концепція Service в Kubernetes використовується для об'єднання декількох Pod, які виконують подібні задачі. Service може бути легко налаштованим для виявлення, горизонтального масштабування та балансування навантаження.

Deployment Controller полегшує набір складних задач управління, таких як:

– масштабування. Програмне забезпечення можу бути розгорнуте перший раз без налаштування масштабування і розгортання можна масштабувати або видаляти в будь-який час;

– видимість. Виявлення завершених, невдалих, а також розготувань в процесі, з можливістю запиту статусу;

– контроль версій. Kubernetes дозволяє оновлювати розгорнуті Pod використовуючи новіші версії імеджів контейнерів з оновленими сервісами або відкочуватися до старіших версій, якщо поточна версія не стабільна.

Серед інших можливостей, Kubernetes спрощує деякі конкретні операції розгортання, які є особливо важливими для розробників сучасних застосувань. Вони включають наступні:

– горизонтальне автоматичне масштабування. Kubernetes автоматично змінює кількість розгорнутих Pod базуючись на використання конкретних ресурсів (у визначених межах);

– розгортання оновлень. Оновлення в Kubernetes організовані в “неперервній манері” через модулі розгортання Pod. Ці неперервні оновлення організовані при роботі з обов'язковими заданими обмеженнями на кількість модулів, які можуть бути недоступними і кількість запасних модулів, які можуть тимчасово існувати;

– розгортання-”канарки”. Корисна схема розгортання нової версії полягає в тому, щоб спочатку протестувати нове розгортання у робочому середовищі, паралельно з попередньою версією і збільшувати масштабування нової версії, одночасно зменшуючи масштабування попередньої версії.

На відміну від традиційних комплексних рішень PaaS, Kubernetes надає широкі можливості для підтримуваних типів застосувань. Він не диктує використання фреймворків чи мови програмування. Kubernetes підтримує широкий спектр робочих навантажень, включаючи робочі навантаження, що не мають статусу, стан і обробку даних. Якщо програма може працювати в контейнері, вона повинна добре працювати на Kubernetes.

Kubernetes знаменує собою прорив для devops, оскільки він дозволяє командам йти в ногу з вимогами сучасної розробки програмного забезпечення. За відсутності Kubernetes, команди часто були змушені створювати скрипти власного розгортання програмного забезпечення, масштабування та оновлення робочих процесів. Деякі організації використовують великі команди для

вирішення цих завдань. Це, безумовно, ефективна модель розробки додатків і сервісів.

2.3.2 Основна мова програмування

Основною мовою програмування була обрана Java. Java має переваги як мова для розробка програмного коду серверної частини.

- розробникам легко вчити мову;
- повністю об'єктно-орієнтована мова програмування;
- мова забезпечує автоматичне очищення пам'яті від об'єктів, що не використовуються, так зване garbage collection, просте управління пам'яттю та такі можливості як узагальнення. Це все робить Java надійною мовою;
- статична типізація на етапі компіляція та перевірка типів в процесі роботи роблять Java безпечною мовою;
- компіляція в байткод дозволяє віртуальній машині JVM виконувати код швидко і на різних платформах.

Багата екосистема підтримує мову. Технологічні гіганти, такі як Oracle, IBM, Google підтримують мову. Велика кількість бібліотек з відкритим кодом, потужних середовищ розробки, інструментів розробки, фреймворків та велика спільнота розробників - додаткові переваги.

2.3.2.1 Масштабованість

В Java розділення модулів дозволяє краще проводити масштабування. Коли збільшуються потреби в обробці операцій читання-запису можна легко додати ресурси і перерозподілити навантаження. Принцип розподілу відповідальності робить цей процес прозорим для програми. Компоненти Java легкодоступні, що полегшує масштабування великих веб-застосунків.

2.3.2.2 Кросплатформність

Однією великою перевагою Java є принцип “Написано один раз - працює скрізь”. Цю особливість також можна назвати портативністю. Скомпільовану

програму Java можна запускати на всіх платформах, що мають відповідну реалізацію JVM. Це твердження правдиве для всіх значних платформ, як Windows, Mac OS та Linux.

2.3.2.3 Управління пам'яттю

Автоматичне управління пам'яттю є великою перевагою Java. Увесь процес повністю автоматизований - так званий “прибиральник сміття” виявляє об'єкти, що більш не використовуються і також групує існуючі об'єкти, щоб звільнити великі послідовні ділянки пам'яті для об'єктів, що її потребують.

2.3.2.4 Багатопоточність

Багатопоточність дозволяє багатьом користувачам одночасно запускати одну програму для своїх індивідуальних задач. Хоча багатопоточність це не новий концепт, Java була першою мовою програмування, яка запропонувала її розробникам.

2.3.3 Допоміжна мова розробки

Kotlin чудово підходить для розробки серверних застосувань, дозволяючи писати короткий та експресивний код і разом з тим підтримувати повну сумісність з існуючими технологічними стеками Java:

2.3.3.1 Експресивність

Інноваційні особливості мови Kotlin, такі як підтримка делеговані властивості і типо-безпечні конструктори, допомагають будувати потужні та прості у використанні абстракції

2.3.3.2 Масштабування

Підтримка корутин в мові програмування Kotlin допомагає будувати сервері сервіси, що масштабуються до масивної кількості клієнтів навіть зі скромним апаратним забезпеченням

2.3.3.3 Сумісність

Kotlin повністю сумісна в усіма Java фреймворками, що дозволяє залишатися на знайомому технологічному стекові і пожинати переваги більш сучасної мови програмування

2.3.3.4 Міграція

Kotlin підтримує поступову міграцію великих баз коду з Java в Kotlin. Можна починати писати нові рядку коду на Kotlin і разом з тим залишати старіші частини системи на Java

2.3.3.5 Інструменти

В додаток до чудової підтримки інтегрованих середовищ розробки, Kotlin пропонує інструменти специфічні для фреймворків (наприклад Spring) як плагіни до середовища Intellij IDEA Ultimate

2.3.3.6 Крива навчання

Для Java розробників почати працювати з Kotlin дуже просто. Автоматичний перетворювач коду Java в Kotlin є частиною плагіну Kotlin в інтегроване середовище розробки

2.3.4 Фреймворк розробки

Spring полегшує розробку корпоративних додатків. Він надає все, що потрібно для використання мови програмування Java в корпоративному середовищі з підтримкою Groovy та Kotlin як альтернативних мов розробки для віртуальної машини JVM, і з гнучкістю, щоб створювати велику кількість різних архітектур, залежно від потреб застосунку. Для версії Spring 5.1, Spring потребує версію JDK 8+ (Java SE 8+) і надає вбудовану підтримку для JDK 11.

Spring передбачає велику кількість сценарії використання застосунків. В великих корпораціях, програми часто існують тривалий час і мають працювати на JDK і програмних серверах, на оновлення яких розробник не може вплинути.

Інші можуть запускати один архів jar на вбудованому сервері, можливо хмарному середовищі. Інші можуть бути відокремленими програмами, що не потребують серверу.

Spring є відкритим програмним забезпеченням. Воно має велику і активну спільноту, що постійно надає відгук базуючись на різноманітних реальних сценаріях використання. Це допомагає Spring успішно розвиватися протягом тривалого часу.

Spring фреймворк поділений на модулі. Програміст може обрати, які модулі він потребує. Серцем є модулі з контейнеру ядра, включаючи конфігураційні моделі і механізми ін'єкції залежностей. Понад того, Spring фреймворк надає значну підтримку для різних архітектур застосунків, включаючи обмін повідомленнями, транзакції та керування сховищами даних і мережеву передачу даних. Він також включає Spring MVC веб-фреймворк, що базується на сервлетах та Spring WebFlux веб-фреймворк для реактивного програмування.

Spring був випущений в 2003 році як реакція на складність ранньої J2EE специфікації. Хоча деякі вважають Java EE та Spring конкурентами, Spring по факту є допоміжним до JEE. Модель програмування Spring не відмовляється від специфікації платформи Java EE, скоріше вона інтегрує дбайливо обрані індивідуальні специфікації Java EE, такі як JPA, JMS, Servlet API та інші. Spring Framework також підтримує специфікації Injection (JSR 330) та Common Annotations (JSR 250), які розробники програм можуть використовувати замість механізмів Spring, передбачених Spring Framework.

З часом роль Java EE в розробці додатків зростала. На початку, Java EE та Spring, додатки були створені для розгортання на сервері додатків. Сьогодні, за допомогою Spring Boot, додатки створюються за допомогою cloud-friendly способу, при цьому з вбудованим контейнером і тривіальні для змін.

Spring продовжує розвиватися. Крім Spring Framework, є й інші проекти, такі як Spring Boot, Spring Security, Spring Data, Spring Cloud, Spring Batch.

Ось основні принципи Spring Framework:

- надання вибору на кожному рівні. Spring дозволяє відкладати дизайн рішення скільки можливо. Наприклад, можливо змінити постачальника послуг збереження даних через конфігурацію без змін в коді. Це ж саме стосується багатьох інших проблем інфраструктури і інтеграції зі сторонніми публічними інтерфейсами програм;
- пристосування до різноманітних перспектив. Spring охоплює гнучкість і не турбується про те, як все має бути зроблено. Він підтримує широкий спектр потреб програми з різних точок зору;
- підтримка зворотної сумісності. Еволюція Spring ретельно контролювалась, щоб привностити лиш невелику кількість серйозних змін між версіями. Spring підтримує ретельно підібраний діапазон версій JDK і сторонніх бібліотек для полегшення обслуговування програм та бібліотек, що залежать від Spring;
- турбота про дизайн API. Команда Spring приділяє багато уваги і часу для створення інтуїтивно зрозумілих інтерфейсів API, які залишаються незмінними протягом багатьох версій та багатьох років;
- високі стандарти якості коду. Spring Framework робить сильний акцент на змістовній, сучасній і точній документації коду. Це один з дуже небагатьох проектів, які можуть претендувати на чисту кодову структуру без кругових залежностей між пакетами.

2.3.5 Автоматизований інструмент CI/CD

Неперервна інтеграція - це практика розробки, при розробники зобов'язані фіксувати зміни у вихідному коді у спільному репозиторії декілька разів в день, або частіше. При кожній зміні проект компілюється і збирається. Це дозволяє командам рано виявляти проблеми. Крім цього, залежно від інструменту неперервної інтеграції, вони можуть надати інші можливості такі

як розгортання сервісу на тестовому сервері, надання інформації про результати збірки та тестування проекту.

Jenkins це інструмент автоматизації з відкритим вихідним кодом написаний на мові програмування Java з плагінами створеними для неперервної інтеграції. Jenkins використовується для неперервної побудови і тестування програмного забезпечення, що полегшує розробникам інтеграцію змін в проекті і отримання користувачами нового білду проекту. Він дозволяє неперервно доставляти програмне забезпечення, інтегруючись з великою кількістю технологій для тестування та розгортання.

З допомогою Jenkins, організації можуть прискорити процес розробки програмного забезпечення за рахунок автоматизації. Jenkins об'єднує процеси життєвого циклу розробки всіх видів, включає компіляцію і побудову, документацію, тестування, архівування, розгортання, статичний аналіз та багато іншого.

Jenkins досягає неперервної інтеграції за допомогою плагінів. Плагіни дозволяють інтеграцію різних етапів DevOps. Існують різні плагіни, наприклад для інтеграції з Git, Slack, Amazon EC2, JUnit, Fortify, SonarQube.

Переваги Jenkins:

- програмне забезпечення з відкритим кодом з великою підтримкою спільноти;
- легкий для встановлення;
- має більш ніж 1000 плагінів для різних задач. Якщо плагін не існує, можна створити свій і поділитися зі спільнотою;
- безкоштовний;
- побудований на мові Java, а тому підтримується всіма популярними платформами;
- популярний інструмент. Зараз понад 147000 компаній по всьому світу користуються Jenkins;

Порівняння процесу розробки до і після використання Jenkins (таблиця 2.6)

Таблиця 2.6 – Порівняння процесів з Jenkins

До використання Jenkins	Після початку використання
Весь вихідний код будується і потім тестується. Тому пошук і виправлення помилок у випадку некоректної роботи збірки є складним процесом і займає багато часу, що в свою чергу уповільнює процес доставки програмного забезпечення	Кожна зміна зроблена у вихідному коді і зафіксована в репозиторії запускає процес компіляції, побудови і тестування проекту. Таким чином, замість перевірки всього коду, розробники можуть зосередитися на певній окремій зміні. Це призводить до більш регулярних випусках програмного забезпечення.
Розробники повинні чекати результатів тестування	Розробники знають результати тестування для кожної зміни в репозиторію коду
Весь процес є ручним	Розробники тільки викладають зміни в центральний репозиторій і Jenkins автоматизує всю іншу частину процесу.

2.4 Аналіз безпеки даних

2.4.1 Kubernetes

Контейнери це ізольоване середовище користувацького оточення, яке часто реалізується за допомогою функцій ядра системи. Кожен контейнер має власний ізольований розділ пам'яті, до якого не мають доступу інші контейнери

та працює в окремому власному процесі. Крім того, середовище можна додатково налаштувати, щоб ізольовувати доступ до нього з апаратного рівня, додати firewall для збільшення безпеки при веб-комунікації та інші засоби безпеки, притаманні користувацьким середовищам та ОС.

2.4.2 REST

Архітектурний підхід для передачі даних REST опирається на протокол прикладного рівня HTTP. Він додає додатковий захист, так як учасники комунікації взаємодіють шляхом передачі переважно текстових даних без безпосереднього контролю на процес виконання коду на іншій стороні. Також домовленість про формати повідомлень, наприклад структура JSON або xml файлів, допомагає виявляти підозрілі та модифіковані повідомлення.

2.4.3 HTTPS

Розширення протоколу HTTP для підтримки шифрування в цілях підвищення безпеки. Дані в протоколі HTTPS передаються поверх криптографічних протоколів SSL або TLS. Він забезпечує захист від атак, побудованих на прослуховуванні мережевого з'єднання - атак прослуховування, атак посередника, при умові, що будуть використовуватись відповідні засоби шифрування.

В TLS використовується як асиметрична схема шифрування (для створення спільного ключа шифрування) так і симетрична (для обміну даними, зашифрованих спільним ключем).

2.4.4 Java

Java краща ніж інші мови програмування в забезпеченні безпеки програм завдяки наступним фактам:

– модель безпеки ізолює програми Java від потенційно шкідливих програм, які користувачі можуть завантажувати з невідомих джерел;

– Java не використовує вказівники, тому неавторизований доступ до блоків пам'яті неможливий;

– з концептом обробки виключень в Java, можна реагувати на серію помилок в процесі роботи програми, що зменшує ризик збою системи;

– різні JVM не виконують код в різному порядку, оскільки Java визначає вс примітиви з певним розміром. Специфікація Java також встановлює порядок виконання операцій;

– віртуальні машини JVM перед запуском тестують байткод кожного разу на виявлення вірусів;

– Java дозволяє розробникам повторно використовувати перевірений код від зовнішніх постачальників, що зменшує ризики;

– мова має механізм контролю доступу для запобігання несанкціонованих запитів від неперевіреного коду;

– Java дозволяє розробникам оголошувати класи як final, від яких ніхто не може наслідуватись. А також зменшувати видимість класів до пакету. Це все зменшує можливе шкідливе використання вразливого коду користувачами бібліотеки.

2.5 Висновки по розділу

Аналіз бізнес процесів, які мають бути реалізованими і автоматизованими системою визначає інструменти, підходи та методики подальшої розробки. Побудовані діаграми процесів роботи сервісів та взаємодії їх з користувачем створені з використанням системи позначень bpmn формально документує задачі описані в попередніх розділах та пришвидшує роботу нових учасників команди, що керуватиме підтримкою продукту.

Був наведений список інструментів та технологій, що були використані при розробці сервісу та їх переваги при виконанні задач для ефективної реалізації системи.

Наведено огляд архітектурного підходу та його застосування до конструювання системи та сервісу генерації звітів.

Наведено огляд засобів безпеки, що включені до інструментів розробки та можливі шляхи її покращення.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обсяг

Тестування буде проведено в декількох точках життєвого циклу розробки продукту. Тестування це дуже “залежна” діяльність. Як результат, тестовий план буде змінюватись протягом життєвого циклу розробки системи. Тестовий план має бути розробленим для кожного рівня тестування продукту.

Тести мають покривати тільки Сервіс Звітності та його взаємодію з зовнішніми сервісами: іншими мікросервісами, БД, джерелами даних, користувачем.

Але не тестувати ці зовнішні сервіси, припускаючи, що вони вже дбайливо протестовані авторами сервісів.

3.2 Об'єкти тестування

Всі елементи, що складають службу звітності, будуть перевірені під час тестування системи.

Вихідний код і запущений сервіс з необхідною версією для тестування будуть наданими розробником сервісу. Розробник сервісу також контролюватиме зміни тестованих версій і повідомлятиме про те, що нові версії служби доступні

Елементами для тестування є пакет com.ams.reporting у проекті AMS та запущений сервіс через через REST API

3.3 Компоненти, що тестуються

- отримання аналітичних даних з джерела даних (БД або мікросервісу);
- перетворення даних домену в моделі звітів (створення звітів Pdf, генерація звітів Excel);

– взаємодія з службою авторизації;

– застосування правильного форматування до структури документів звіту;

– відповідь з кодом помилки на запити з поганим токеном авторизації;

– стратегія реагування на помилкові ситуації, коли не вдалося підключитися до необхідного зовнішнього сервісу;

– продуктивність;

– час відгуку сервісу.

3.4 Компоненти, що не тестуються

– безпека мікросервісу;

– транспортні інтерфейси;

– мікросервісна інфраструктура;

– мікросервісне середовище;

– правильна робота зовнішніх сервісів;

– правильна робота БД, що використовується для доступу до даних;

– ці функції не будуть перевірені, оскільки розробники та спеціалісти з тестування повинні гарантувати зовнішні якість сервісів, фреймворків, контейнера веб додатків, бази даних та транспортного протоколу.

3.5 Підхід

Тестування - це процес аналізу елемента програмного забезпечення для виявлення відмінностей між існуючими та необхідними умовами та оцінки особливостей елемента програмного забезпечення.

Якість мікросервісу звітності повинна бути виявленою за допомогою різних видів тестування. Окремі частини (класи) компонентної структури повинні бути покриті компонентними тестами з заданим покриттям. Також необхідно створити деякі інтеграційні тести для тестування сумісність

компонентів. Тестування продуктивності слід проводити для виявлення обмежень. Для перевірки взаємодії мікросервісу зі своїми клієнтами слід проводити тестування методом чорного ящика.

3.5.1 Тестування компонентів

Перевірка авторизацію користувача
Отримання даних для звітів з джерела даних
Перетворення даних домену у формат даних звіту
Визначення структури документа звітів (Excel, pdf, xml)

3.5.2 Тестування інтеграції

Створення звітів у форматі PDF
Створення звітів у форматі Excel
Помилки, коли користувач має неправильні повноваження

3.5.3 Тестування інтерфейсу

Кінцеві точки API

3.5.4 Регресійне тестування

Виконане вручну або юніт тестами

3.6 Критерії проходження тестів

Система повинна задовольняти наступним вимогам

- використовувана память для мікросервісу, включаючи JVM, не повинна перевищувати 800 МБ ОЗУ;
- мікросервіс запобігає використання неавторизованими користувачами функціями звітування;
- час відгуку служби має бути менше 900 мс;

– сумісність процедур користувача з іншим клієнтським обладнанням та програмним забезпеченням повинна задовольняти клієнта;
– немає помилок з пріоритетом вище, ніж High;
– покриття юніт тестами більше 70%;

3.7 Результати проведення тестування

– тест план;
– репорт про проходження юніт тестів;
– репорт по покриттю юніт тестами;
– специфікація тест кейсів;
– репорт тестових інцидентів;
– логи до репорту тестових інцидентів;
– узагальнюючий репорт по тестуванню.

3.8 Задачі для проведення тестування

– підготувати план тестування;
– підготувати технічні характеристики тестування;
– розробити модульні тести;
– розробити API тести;
– розробити навантажувальні тести;
– розробити автоматизовані юніт тести;
– підготувати специфікацію ручного тесту;
– підготувати звіт про підсумки компонентного тестування;
– розробити автоматизовані тести для перевірки навантаження;
– підготувати звіт про результати навантажувального тестування;
– розібрати елементи звіту про інциденти.

3.9 Технічні потреби

– Ubuntu OS 14.04, 16.04;

– інструменти CI/CD (Jenkins LTS 2.150.1) ;
– Java (JDK 1.8+);
– Kotlin (1.2.71+);
– RAM 4 GB;
– частота CPU 1.0 GHz;
– зовнішня память 8 GB;
– підключення до інтернету (закачування 20 Mbps, вивантаження 10 Mbps).

3.10 Контрольний приклад

Таблиця 3.1 – Перевірка можливості створення звіту у форматі xml

Мета тесту	Перевірка можливості створення звіту у форматі xml
Початковий стан	Розгорнуті сервіс генерації репортів і вимірів
Вхідні дані	Параметри вибірки вимірів.
Схема проведення тесту	Надіслати запит на кінцеву точку /reports/xml з необхідними параметрами
Очікуваний результат	Створено репорт, який знаходиться у відповіді з сервісу, перелік даних відповідає схемі звіту.
Стан програмного продукту після проведення випробувань	Створено репорт, який знаходиться у відповіді з сервісу, перелік даних відповідає схемі звіту.

Таблиця 3.2 – Перевірка можливості створення звіту у форматі excel

Мета тесту	Перевірка можливості створення звіту у форматі excel
Початковий стан	Розгорнуті сервіс генерації репортів і вимірів
Вхідні дані	Параметри вибірки вимірів.

Продовження таблиці 3.2

Схема проведення тесту	Надіслати запит на кінцеву точку /reports/excel з необхідними параметрами
Очікуваний результат	Створено репорт у форматі табличного документу, який знаходиться у відповіді з сервісу, перелік даних відповідає схемі звіту.
Стан програмного продукту після проведення випробувань	Створено репорт у форматі табличного документу, який знаходиться у відповіді з сервісу, перелік даних відповідає схемі прикладу.

Таблиця 3.3 – Перевірка можливості створення звіту у форматі pdf

Мета тесту	Перевірка можливості створення звіту у форматі pdf
Початковий стан	Розгорнуті сервіс генерації репортів і вимірів
Вхідні дані	Параметри вибірки вимірів.
Схема проведення тесту	Надіслати запит на кінцеву точку /reports/pdf з необхідними параметрами
Очікуваний результат	Створено репорт у форматі pdf, який знаходиться у відповіді з сервісу, перелік даних відповідає схемі звіту.
Стан програмного продукту після проведення випробувань	Створено репорт у форматі pdf, який знаходиться у відповіді з сервісу, перелік даних відповідає схемі прикладу.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання версії сервісу необхідно створити нову задачу за допомогою інструменту автоматизації неперервної доставки Jenkins з цьолою головною гілкою в центральному репозиторії коду.

Задача створить пайплан з декількох кроків, такі завантаження нової версії коду, компіляція, збірка, розгортання в Docker контейнері, що керується Kubernetes та юніт тестування. На рисунку 3.1 візуальне зображення консолі Jenkins

Рисунок 4.1 – Графічна консоль виконання Jenkins

Якщо якісь з кроків завершився з помилкою варто відкрити логи кроку, виявити помилку, виправити, викласти зміни в центральний репозиторій та повторити спробу.

Якщо всі кроки пройшли успішно, сервіс стає доступним для користування. Перевірити стан контейнеру та середовища, кількість активних копій сервісу можна в консолі Kubernetes, що надає графічний користувацький інтерфейс

(рисунок 4.2).

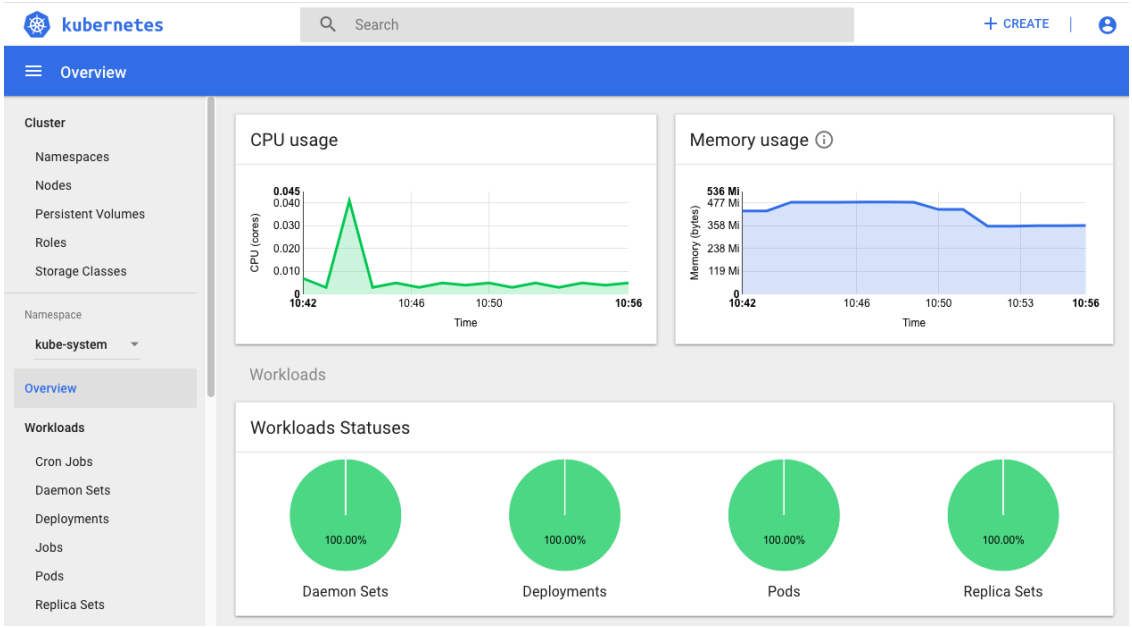


Рисунок 4.2 – Консоль Kubernetes

При виникненні помилок в роботі сервісу, можна переглянути логи (рисунок 4.3), що створює копія сервісу та виявити причину.

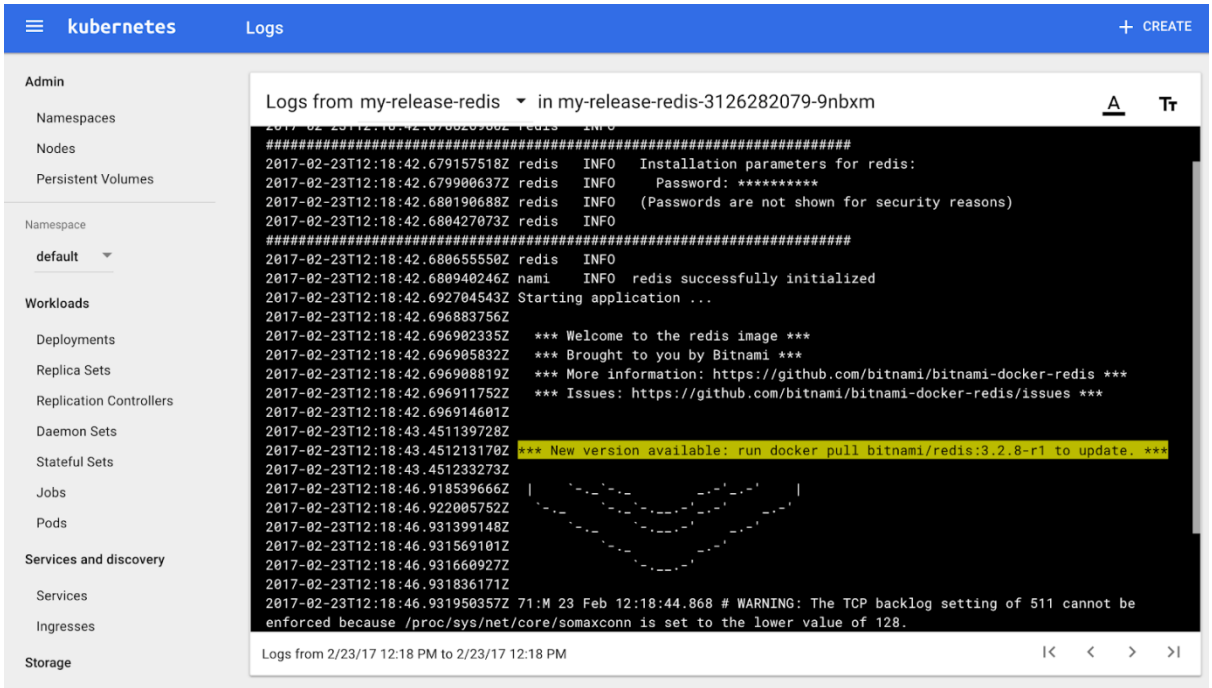


Рисунок 4.3 – Консоль Kubernetes з логами

Після цього потрібно виправити проблему, викласти зміни в центральний репозиторій та створити задачу в Jenkins як описано вище.

Копія сервісу з помилкою буде замінена на нову з виправленнями.

4.2 Робота з програмним забезпеченням

Інструкція для роботи з програмним забезпеченням наведена в документі «Керівництво користувача»

ВИСНОВКИ

В ході роботи над диплом проектом було проведено аналіз проблем грунту при ведення сільського господарства пов'язаних з його щільністю, шляхи виявлення цих проблем та їх вплив на ефективність використання полів. Та обгрунтована необхідність створення геоінформаційної системи для цих даних.

Було розглянуто поняття геоінформаційної системи, шляхи використання просторових даних для задач повязаних з діяльністю фермерів. Оглянуті компоненти геоінформаціних систем. Також проаналізовано основні сценарії їх використання.

Виконано порівняння використання принципів мікросервісної архітектури до монолітної архітектури враховуючи вимоги поставлені даною системою. Створено компонентну діаграму сервісу та діаграму системи та залежностей її окремих сервісів.

Обрано засоби та інструменти розробки (мови програмування Java та Kotlin, фреймворк Spring, Jenkins для автоматизації перевірки, побудови та розгортання програмного забезпечення та Kubernetes для керування контейнерами з запущеними програмами, незалежно від обраної платформи).

Був створений мікросервіс для генерації звітів у форматах xml, табличного документу та pdf документу. Сервіс реалізований з використанням принципів чистої архітектури.

Код скомпільований, зібраний та розгорнутий на власному сервері за допомогою інструменту Jenkins.

Написана інструкція користувача та план тестування.

ПЕРЕЛІК ПОСИЛАНЬ

1) GIS Enabling Smart Agriculture [Електронний ресурс]: (Стаття) / ResearchGate – Електрон.дан. (1 файл) – 2015. – Режим доступу: https://www.researchgate.net/publication/281295193_GIS_Enabling_Smart_Agriculture – Назва з екрана

2) Use of GIS in Argriculture [Електронний ресурс]: (Стаття) / SmallFarms – Електрон.дан. (1 файл) – 2017. – Режим доступу: https://smallfarms.cornell.edu/2017/04/03/use-of-gis/ – Назва з екрана

3) Reasons to use Kubernetes [Електронний ресурс]: (Стаття) / InfoWorld – Електрон.дан. (1 файл) – 2017. – Режим доступу: https://www.infoworld.com/article/3173266/4-reasons-you-should-use-kubernetes.html – Назва з екрана

4) Concepts of Kubernetes [Електронний ресурс]: (Стаття) / Kubernetes – Електрон.дан. (1 файл) – 2019. – Режим доступу: https://kubernetes.io/docs/concepts/– Назва з екрана

5) Spring Framework Overview [Електронний ресурс]: (Стаття) / Spring – Електрон.дан. (1 файл) – 2019. – Режим доступу: https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html – Назва з екрана

6) The Clean Architecture [Електронний ресурс]: (Стаття) / Clean Coder – Електрон.дан. (1 файл) – 2012. – Режим доступу: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html – Назва з екрана

7) The Clean Architecture [Електронний ресурс]: (Стаття) / Clean Coder – Електрон.дан. (1 файл) – 2012. – Режим доступу: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html – Назва з екрана

8) Kubernetes on Ubuntu [Електронний ресурс]: (Стаття) / How to forge – Електрон.дан. (1 файл) – 2018. – Режим доступу: https://www.howtoforge.com/tutorial/how-to-install-kubernetes-on-ubuntu/ - Назва з екрана

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства

Технічне завдання

КПІ.ІП-5103.045440.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____ Д.С. Смаковський

Нормоконтроль:

_____ М.М. Головченко

Виконавець:

_____М.В. Гарбовський

Київ – 2019 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства

Галузь застосування: Web-додатки, клієнти для доступу до Internet-ресурсів

Наведене технічне завдання поширюється на розробку програмного забезпечення Мікросервіс генерації репортів для хмарної мікросервісервісної геоінформаційної системи для сільського господарства [045440], котра використовується для генерації звітів та призначена для сільськогосподарських підприємств, що потребують геоінформаційну систему для контролю щільності ґрунту.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки Мікросервісу генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для фермерів та сільськогосподарських підприємств, що прагнуть автоматизувати процеси керування, обробки та представлення просторових даних про щільність ґрунту на різній глибині і одночасно забезпечити високий рівень захисту даних від неконтрольованого та несанкціонованого доступу.

Метою розробки є побудова мікросервісів геоінформаційної хмарної системи для генерації репортів та керування доступом до вимірів про щільність ґрунту.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

- аутентифікація;
- додавання даних про виміри;
- додавання даних про компанії;
- додавання даних про поля;
- додавання даних про пристрої;
- можливість кастомізованих запитів на отримання даних про виміри щільності;
- генерація статистичного звіту про виміри щільності грунту для певного поля за вказаний період у форматі xml;
- генерація статистичного звіту про виміри щільності грунту для певного поля за вказаний період у форматі табличного документу;
- генерація статистичного звіту про виміри щільності грунту для певного поля за вказаний період у форматі pdf документу.

4.1.2 Розробку виконати на платформі Unix

4.1.3 Додаткові вимоги:

- генерація звітів українською мовою;
- можливість розширення списку підтримуваних мов.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Обслуговування

4.3.3 Обслуговуючий персонал

Група тестування має пройти тренінги по наступним технологіям та підходам

- Введення до архітектури системи мікросервісів;
- Spring Boot;
- Spring Security;
- Hibernate;
- OAuth2;
- RESTful services.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2 Мінімальна конфігурація технічних засобів:

Unix OS 14.04, 16.04,

AWS,

Git,

Інструменти CI/CD (Jenkins LTS 2.150.1),

Java (JDK 1.8+),

Kotlin (1.2.71+),

RAM 4 GB,

Частота CPU 1.0 GHz,

Зовнішня память 8 GB

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Unix.

4.5.2 Вхідні дані повинні бути представлені в наступному форматі:

4.5.2.1 Генерація звіту у форматі xml

Назва: reports/xml

Метод: GET

Заголовки: Authentication Bearer: “значення access token”

Параметри:

fieldId - ідентифікатор поля

dateFrom - дата початку вимірювань у форматі дд.мм.рр

dateTo - дата закінчення вимірювань у форматі дд.мм.рр

unit - одиниця вимірювань (kPa)

Приклад:

localhost:8080/reports/xml?fieldId=some_field_id&dateFrom=12.10.19&dateTo=20.10.19&unit=kPa

4.5.2.2 Генерація звіту у форматі табличного документу

Назва: reports/excel

Метод: GET

Заголовки: Authentication Bearer: “значення access token”

Параметри:

fieldId - ідентифікатор поля

dateFrom - дата початку вимірювань у форматі дд.мм.рр

dateTo - дата закінчення вимірювань у форматі дд.мм.рр

unit - одиниця вимірювань (kPa)

Приклад:

localhost:8080/reports/excel?fieldId=some_field_id&dateFrom=12.10.19&dateTo=20.10.19&unit=kPa

4.5.2.3 Генерація звіту у форматі документу pdf

Назва: reports/pdf

Метод: GET

Заголовки: Authentication Bearer: “значення access token”

Параметри:

fieldId - ідентифікатор поля

dateFrom - дата початку вимірювань у форматі дд.мм.рр

dateTo - дата закінчення вимірювань у форматі дд.мм.рр

unit - одиниця вимірювань (kPa)

Приклад:

localhost:8080/reports/pdf?fieldId=some_field_id&dateFrom=12.10.19&dateTo=20.10.19&unit=kPa

4.5.3 Результати повинні бути представлені в наступному форматі:

4.5.3.1 Генерація звіту у форматі xml

Статус код: 200

Заголовки:

Content type: text/xml

Тіло відповіді – xml звіт

4.5.3.2 Генерація звіту у форматі табличного документу

Статус код: 200

Заголовки:

Content type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet

Content-Disposition: inline; filename=device-report_24.12.19_29.12.19.xlsx

Тіло відповіді – табличний документ

4.5.3.3 Генерація звіту у форматі pdf документу

Статус код: 200

Заголовки:

Content type: application/x-pdfContent type: text/xml

Content-Disposition: inline; filename=device-report_24.12.19_29.12.19.pdf

Expires: 0

Cache-Control: must-revalidate, post-check=0, pre-check=0

Pragma: public

Тіло відповіді – pdf документ

4.5.4 Засоби розробки

Система автоматичного розгортання: Kubernetes

Головна мова програмування: Java

Допоміжна мова програмування: Kotlin

Фреймворк розробки: Spring Framework

Фреймворк тестування: JUnit

Система керування версіями: Git

Система керування репозиторіями: GitLab

Система керування проектом: Jira

Інструмент безперервної інтеграції: Jenkins

Архітектурний підхід системи: мікросервісна архітектура

Архітектурний підхід сервісу: Clean Architecture

Архітектура веб-комунікації: REST

Інтегроване середовище розробки: Intellij IDEA Community Edition

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 60 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Керівництво системного програміста

5.3.5 Програма та методика тестування

5.4 Графічна частина повинна бути виконана на листі формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема структура інформаційної системи.

5.4.2 Схема функціональна програмного забезпечення.

5.4.3 Схема структурна варіантів використання

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

	Назва етапу	Строк,	Звітність
.	Вивчення літератури за тематикою проекту	20.04.19	
.	Розробка технічного завдання	23.04.19	Технічне завдання
.	Аналіз вимог та уточнення специфікацій	25.04.19	Специфікації програмного забезпечення
.	Проектування структури програмного забезпечення, проектування компонентів	28.04.19	Схема структурна програмного забезпечення та специфікація компонентів
.	Програмна реалізація програмного забезпечення	10.05.19	Тексти програмного забезпечення
.	Тестування програмного забезпечення	12.05.19	Тести, результати тестування
.	Розробка матеріалів текстової частини проекту	13.05.19	Пояснювальна записка.
.	Розробка матеріалів графічної частини проекту	14.05.19	Графічний матеріал проекту
.	Оформлення технічної документації проекту	20.05.19	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ О.А. Павлов
“___” _____ 2019 р.

Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства

Програма та методика тестування

КПІ.ІП-5103.045440.04.51

Київ – 2019

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Всі елементи, що складають службу звітності, будуть перевірені під час тестування системи.

Вихідний код і запущений сервіс з необхідною версією для тестування будуть наданими розробником сервісу. Розробник сервісу також контролюватиме зміни тестованих версій і повідомлятиме про те, що нові версії служби доступні

Елементи для тестування такі:

- пакет com.ams.reporting у проекті AMS;
- Запущений сервіс через через REST API.

2 МЕТА ТЕСТУВАННЯ

Процес тестування має на меті наступні цілі:

- забезпечення належного рівня безпеки даних;
- відповідність функціональним вимогам;
- перевірка коректності генерації репортів;
- перевірка належної роботи сервісу при помилках сторонніх сервісів.

3 МЕТОДИ ТЕСТУВАННЯ

Використовуються наступні методи тестування:

- тестування інтерфейсу;
- юніт-тестування;
- ручне тестування;
- тестування продуктивності.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

4.1 Тестування компонентів

Перевірка авторизації користувача

Отримання даних для звітів з джерела даних

Перетворення даних домену у формат даних звіту

4.2 Тестування інтеграції

Створення звітів у форматі PDF

Створення звітів у форматі Excel

Помилки, коли користувач має неправильні повноваження

4.3 Тестування інтерфейсу

Кінцеві точки API

4.4Тестування відновлення

Перевірити стратегію відновлення після помилок, коли зв'язок із зовнішніми сервісами втрачено

4.4Регресійне тестування

Виконане вручну або юніт тестами

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ О.А. Павлов
“___” _____ 2019 р.

Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства

Керівництво користувача

КПІ.ІП-5103.045440-05-34

“ПОГОДЖЕНО”
Керівник проекту:
___ _____ Д.С. Смаковський

Нормоконтроль:
_____ М.М. Головченко

Виконавець:
___ _____ М.В. Гарбовський

Київ – 2019 року

ВЗАЄМОДІЯ ІЗ ПЛАТФОРМОЮ

Мікросервіс генерації репортів має три веб точки входу для кожного формату звітів.

Таблиця 1 Опис точки входу для генерації xml репорту

Назва	/reports/xml
Метод	GET
Заголовки	Authentication Bearer: “значення access token”
Параметри	fieldId – ідентифікатор поля dateFrom – дата початку вимірювань у форматі дд.мм.рр dateTo – дата закінчення вимірювань у форматі дд.мм.рр unit – одиниця вимірювань (kPa)

Для генерації звіту у форматі xml потрібно послати запит на кінцеву точку описану в таблиці 1

```
<report>
  <field id="12">
    <name>field-name-12</name>
    <square>46.329297680921655</square>
    <groups>
      <group>field-group-1</group>
      <group>field-group-2</group>
      <group>field-group-3</group>
      <group>field-group-4</group>
    </groups>
    <coordinates>
      <point>
        <lat>20.0</lat>
        <lon>30.0</lon>
      </point>
      <point>
        <lat>22.0</lat>
        <lon>33.0</lon>
      </point>
      <point>
        <lat>20.0</lat>
        <lon>33.0</lon>
      </point>
      <point>
        <lat>22.0</lat>
        <lon>30.0</lon>
      </point>
    </coordinates>
    <devices>
      <device id="device-12">
        <responsible_users>
          <user>
            <first_name>John</first_name>
            <last_name>Smith</last_name>
          </user>
          <user>
            <first_name>Bryan</first_name>
            <last_name>Turner</last_name>
          </user>
        </responsible_users>
        <measures>
          <measure id="1">
            <location>
              <lat>20.0</lat>
              <lon>21.0</lon>
            </location>
            <date>
              <day>16</day>
              <month>5</month>
              <year>2019</year>
            </date>
            <time>
              <hours>14</hours>
              <minutes>14</minutes>
              <seconds>47</seconds>
            </time>
            <type>type1</type>
            <values>
              <value>
                <depth>0.0</depth>
                <density>3918.425280797864</density>
              </value>
              <value>
                <depth>2.5</depth>
                <density>2065.1979194739533</density>
              </value>
              <value>
                <depth>5.0</depth>
                <density>4866.854984532714</density>
              </value>
              <value>
                <depth>7.5</depth>
                <density>398.32135994047445</density>
              </value>
              <value>
                <depth>10.0</depth>
                <density>2046.8135391315934</density>
              </value>
              <value>
                <depth>12.5</depth>
                <density>2234.8940581830593</density>
              </value>
              <value>
                <depth>15.0</depth>
                <density>3910.5112327260867</density>
              </value>
            </values>
          </measure>
        </measures>
      </device>
    </devices>
  </field>
  <unit>kPa</unit>
</report>
```

Рисунок 1 Приклад репорту у форматі xml

Приклад відповіді – репорту у форматі xml – зображено на рисунку 1

Таблиця 2 Опис точки входу для генерації excel репорту

Назва	/reports/excel
Метод	GET
Заголовки	Authentication Bearer: “значення access token”

Продовження таблиці 2

Параметри	fieldId - ідентифікатор поля dateFrom - дата початку вимірювань у форматі дд.мм.рр dateTo - дата закінчення вимірювань у форматі дд.мм.рр unit - одиниця вимірювань (kPa)

Для генерації звіту у форматі excel потрібно послати запит на кінцеву точку описану в таблиці 2.

Рисунок 2 Перший аркуш репорту excel

Згенерований репорт складається з двох аркушів. Перший аркуш зображений на рисунку В.2. Він містить дані про поле, пристрій вимірювання, працівників. що здійснювали вимірювання, легенду про діапазони щільності для кожного типу грунту та кольорове позначення залежно від значення щільності, список вимірів щільності для кожної точки та різних глибинах з забарвленням, що відповідає типу грунту, статистичні дані про середні, мінімальні та максимальні значення на різних значеннях глибини.

Рисунок 3 Другий аркуш репорту excel

Наступний аркуш (рисунок 3) містить значення щільності та графіки залежностей щільності грунту залежно від глибини вимірювань.

Таблиця 3 Опис точки входу для генерації pdf репорту

Назва	/reports/pdf
Метод	GET
Заголовки	Authentication Bearer: “значення access token”
Параметри	fieldId - ідентифікатор поля dateFrom - дата початку вимірювань у форматі дд.мм.рр dateTo - дата закінчення вимірювань у форматі дд.мм.рр unit - одиниця вимірювань (kPa)

Для генерації звіту у форматі excel потрібно послати запит на кінцеву точку описану в таблиці 3. Pdf репорт складається з багатьох аркушів.

Компанія:AgriCompany 12
Поле field-name-12 (Площа 0.06 га)
Група полів, якої належить обране поле
field-group-1
field-group-2
field-group-3
field-group-4
Обраний період: 12.10.18-15.10.18

Відповідальні особи за обраний період:
John-Smith
Bryan-Turner

Рисунок 4 Перший аркуш репорту pdf

Перший аркуш містить дані про поле, пристрій вимірювання, працівників. що здійснювали вимірювання.

Тип грунту:	Пухкий			Щільний			Переущільнений		
Щільність	0-950	950-1400	1400-1750	1750-1900	1900-2300	2300-2900	2900-3500	3500-4000	4000-6000

№	1	2	3	4	5	6	7	8	9	10
Дата	18.05.19	16.05.19	18.05.19	20.05.19	22.05.19	18.05.19	18.05.19	18.05.19	18.05.19	18.05.19
Час	07:22:18	07:22:18	07:22:18	07:22:18	07:22:18	07:22:18	07:22:18	07:22:18	07:22:18	07:22:18
ID пристрою	device-12	device-12	device-12	device-12	device-12	device-12	device-12	device-12	device-12	device-12
Координати	Google map	Google map	Google map	Google map	Google map	Google map	Google map	Google map	Google map	Google map
Наконечник	first	type1	type2	type3	type4	type2	type2	type2	type2	type2
Глибина(см)	Щільність									
0.0	1794	4001	3993	4493	3557	589	1723	4374	1377	800
2.5	2401	3746	75	577	38	2096	2261	3917	3270	4246
5.0	1478	2708	584	4319	3234	3206	4470	3520	3030	135
7.5	4590	727	2091	1727	4816	1609	4255	3768	2574	4072
10.0	2836	4671	2997	1161	1108	1973	4503	3942	3083	3037
12.5	4265	4458	2182	4425	164	1151	4499	865	3952	1357
15.0	3154	4909	4294	733	4065	1018	1755	4639	1182	921
17.5	2607	82	1679	1265	2144	2404	2389	3663	965	1401
20.0	1130	2504	4411	1838	2218	2821	3661	1460	586	2222
22.5	1176	992	871	2494	2698	4383	1672	3215	3103	3091
25.0	2055	2022	3901	2184	2224	4503	861	1746	1909	2030
27.5	2628	908	3756	2098	1158	71	3106	2931	4547	78
30.0	1015	524	1676	4044	2107	1894	4933	2436	4666	4256
32.5	4564	1695	4222	328	514	4846	3870	2639	2282	4586
35.0	1148	1393	1545	1264	2358	4627	283	4754	343	1538
37.5	4124	2114	945	12	4346	2343	706	1489	955	3929
40.0	74	2797	2904	835	1714	4125	790	4194	4497	2452
42.5	720	3036	2850	404	4277	2906	1470	553	2427	1133
45.0	2276	3371	1110	2698	729	2922	4780	2800	4080	3030

Рисунок 5 Другий аркуш репорту pdf

Другий аркуш (рисунок 5) містить легенду про діапазони щільності для кожного типу грунту та кольорове позначення залежно від значення щільності.

Другий і наступні аркуші містять список вимірів щільності для кожної точки та різних глибинах з забарвленням, що відповідає типу грунту, статистичні дані про середні, мінімальні та максимальні значення на різних значеннях глибини.

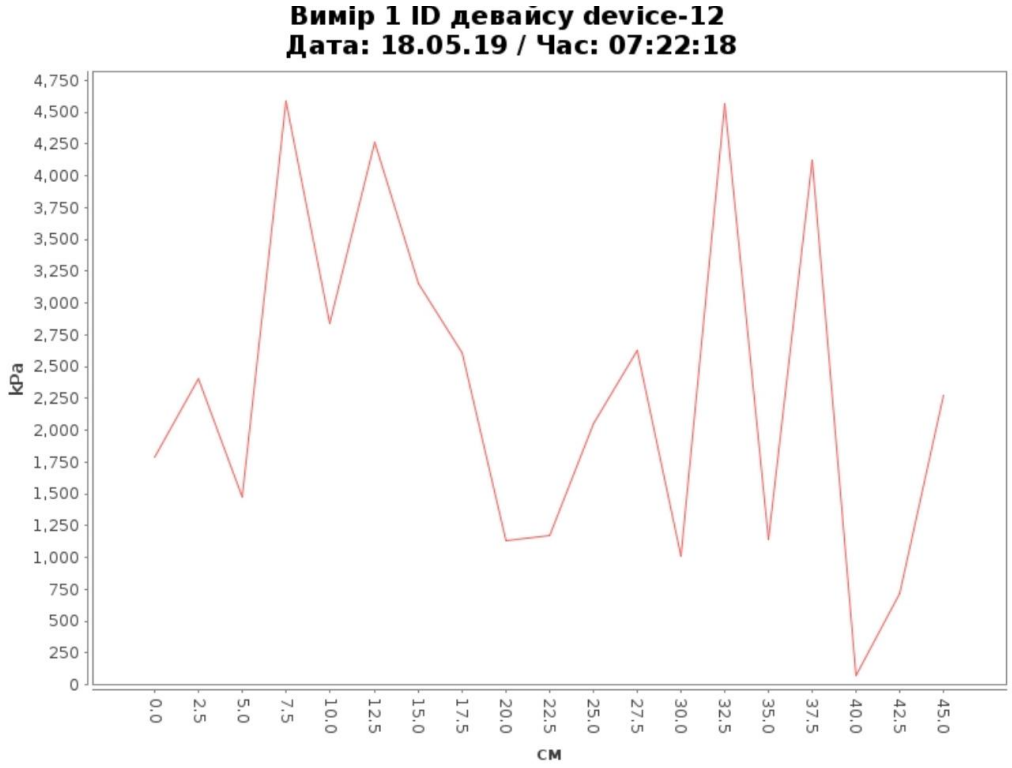


Рисунок 6 аркуш репорту pdf з графіком

Після таблиці вимірів аркуші містять графіки залежностей щільності грунту залежно від глибини вимірювань. Приклад аркушу зображений на рисунку 6.

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ О.А. Павлов
“___” _____ 2019 р.

Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства

Опис програми

КПІ.ІП-5103.045440-06-13

“ПОГОДЖЕНО”
Керівник проекту:
___ Д.С. Смаковський

Нормоконтроль:
_____ М.М. Головченко

Виконавець:
___ М.В. Гарбовський

Київ – 2019 року

Тексти програмного коду

Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарствагеоданими

(Найменування програми (документа))

DVD-R

(Вид носія даних)

123 арк, 531 Кб

(Обсяг програми (документа) , арк.,)

Київ - 2019

```
package agriculture.controller.mapper;

import agriculture.domain.InputReportRequest;
import agriculture.domain.Unit;
import agriculture.utils.DateParserKt;
import org.springframework.stereotype.Component;

import java.time.LocalDate;

@Component
public class InputReportRequestMapper {

    public InputReportRequest toRequest(
        String fieldId,
        String dateFromString,
        String dateToString,
        String unit,
        String deviceId
    ) throws InvalidRequestException {
        LocalDate dateFrom = DateParserKt.parseDate(dateFromString);
        LocalDate dateTo = DateParserKt.parseDate(dateToString);

        return new InputReportRequest(fieldId, deviceId, dateFrom, dateTo, parseUnit(unit));
    }

    private Unit parseUnit(String unit) {
        try {
            return Unit.valueOf(unit.toUpperCase());
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
            return Unit.OTHER;
        }
    }

}


package agriculture.controller.mapper;

public class InvalidRequestException extends Exception {
    public InvalidRequestException(String message) {
        super(message);
    }
}

package agriculture.controller;

import agriculture.controller.mapper.InvalidRequestException;
import agriculture.domain.ApplicationException;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
```

```
import java.io.IOException;

@ControllerAdvice
public class GlobalExceptionHandler {

   @ExceptionHandler(InvalidRequestException.class)
   public ResponseEntity<String> handleInvalidRequestException(Exception e) {
      return ResponseEntity.badRequest().body(e.getMessage());
   }

   @ExceptionHandler(value = {IOException.class, ApplicationException.class})
   public ResponseEntity<String> handleReportGenerationException(Exception e) {
      return ResponseEntity.status(500).body(e.getMessage());
   }
}

package agriculture.controller;

import agriculture.controller.mapper.InputReportRequestMapper;
import agriculture.controller.mapper.InvalidRequestException;
import agriculture.domain.ApplicationException;
import agriculture.domain.InputReportRequest;
import agriculture.domain.PreparedReportData;
import agriculture.domain.interactor.PrepareReportDataInteractor;
import agriculture.presentation.document.DocumentPresenter;
import agriculture.presentation.document.Labels;
import agriculture.presentation.document.View;
import agriculture.presentation.document.excel.CellStyleProvider;
import agriculture.presentation.document.excel.ChartCreator;
import agriculture.presentation.document.excel.ExcelHelper;
import agriculture.presentation.document.excel.ExcelView;
import agriculture.presentation.document.mapper.DocumentReportDataMapper;
import agriculture.presentation.document.pdf.PdfView;
import com.itextpdf.text.DocumentException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@RestController
public class ReportController {

   private final PrepareReportDataInteractor reportDataInteractor;
   private final InputReportRequestMapper mapper;
   private final ReportPresenter xmlPresenter;

   @Autowired
```

```
    public ReportController(
            PrepareReportDataInteractor reportDataInteractor,
            InputReportRequestMapper mapper,
            @Qualifier("XML_PRESENTER")
            ReportPresenter xmlPresenter
    ) {
        this.reportDataInteractor = reportDataInteractor;
        this.mapper = mapper;
        this.xmlPresenter = xmlPresenter;
    }

    @GetMapping(value = "/reports/xml")
    public void createXmlFieldReport(
        @RequestParam String fieldId,
        @RequestParam String dateFrom,
        @RequestParam String dateTo,
        @RequestParam String unit,
        @RequestParam(required = false) String deviceId,
        HttpServletResponse response
    ) throws IOException, InvalidRequestException, ApplicationException {
        handleReportRequest(fieldId, dateFrom, dateTo, unit, deviceId, xmlPresenter, response);
    }

    @GetMapping(value = "/reports/excel")
    public void createExcelFieldReport(
            @RequestParam String fieldId,
            @RequestParam String dateFrom,
            @RequestParam String dateTo,
            @RequestParam String unit,
            @RequestParam(required = false) String deviceId,
            HttpServletResponse response
    ) throws IOException, InvalidRequestException, ApplicationException {
        handleReportRequest(fieldId, dateFrom, dateTo, unit, deviceId,
prepareDocumentPresenter(prepareExcelView()), response);
    }

    @GetMapping(value = "/reports/pdf")
    public void createPdfFieldReport(
            @RequestParam String fieldId,
            @RequestParam String dateFrom,
            @RequestParam String dateTo,
            @RequestParam String unit,
            @RequestParam(required = false) String deviceId,
            HttpServletResponse response
    ) throws IOException, DocumentException, InvalidRequestException, ApplicationException {
        handleReportRequest(fieldId, dateFrom, dateTo, unit, deviceId, prepareDocumentPresenter(new
PdfView()), response);
    }

    private static View prepareExcelView() {
        CellStyleProvider cellStyleProvider = new CellStyleProvider();
        ExcelHelper helper = new ExcelHelper(cellStyleProvider);
```

```
        ChartCreator chartCreator = new ChartCreator();
        return new ExcelView(cellStyleProvider, helper, chartCreator);
    }

    private static ReportPresenter prepareDocumentPresenter(View view) {
        DocumentReportDataMapper mapper = new DocumentReportDataMapper(Labels.INSTANCE);
        return new DocumentPresenter(view, mapper);
    }

    private void handleReportRequest(
            String fieldId,
            String dateFrom,
            String dateTo,
            String unit,
            String deviceId,
            ReportPresenter reportPresenter,
            HttpServletResponse response
    ) throws IOException, ApplicationException, InvalidRequestException {
        InputReportRequest reportRequest = mapper.toRequest(fieldId, dateFrom, dateTo, unit, deviceId);

        PreparedReportData preparedReportData = reportDataInteractor.prepare(reportRequest);

        reportPresenter.format(preparedReportData, response);
    }
}
```

```
package agriculture.controller

import agriculture.domain.ApplicationException
import agriculture.domain.PreparedReportData
import java.io.IOException
import javax.servlet.http.HttpServletResponse

interface ReportPresenter {
    @Throws(ApplicationException::class, IOException::class)
    fun format(preparedReportData: PreparedReportData, response: HttpServletResponse)
}
```

```
package agriculture.data

import agriculture.domain.entity.*
import agriculture.domain.repository.*
import org.springframework.stereotype.Component
import java.time.LocalDate
import java.time.LocalDateTime
import kotlin.random.Random

@Component
class MemoryFieldRepository: FieldRepository {
    override fun getById(fieldId: String): Field =
        Field(
            id = fieldId,
```

```
            name = "field-name-$fieldId",
            square = 100 * Random.nextDouble(),
            groups = fieldGroups(),
            coordinates = coordinates()
        )

    private fun fieldGroups() =
        listOf(
            FieldGroup("1", "field-group-1"),
            FieldGroup("2", "field-group-2"),
            FieldGroup("3", "field-group-3"),
            FieldGroup("4", "field-group-4")
        )

    private fun coordinates() =
        listOf(
            Location(20.0, 30.0),
            Location(22.0, 33.0),
            Location(20.0, 33.0),
            Location(22.0, 30.0)
        )
}

@Component
class MemoryCompanyRepository: CompanyRepository {
    override fun getByFieldId(fieldId: String): Company =
        Company(
            id = "company-$fieldId",
            name = "AgriCompany $fieldId"
        )
}

@Component
class MemoryDeviceRepository: DeviceRepository {

    override fun getByDeviceId(deviceId: String): Device =
        Device(
            id = "device-$deviceId",
            code = "device-code-${deviceId.length}",
            inventoryNumber = "inventory-$deviceId",
            description = "Some device"
        )

    override fun getByFieldId(field: String, dateFrom: LocalDate, dateTo: LocalDate): List<Device> =
        listOf(
            Device(
                id = "device-$field",
                code = "device-code-${field.length}",
                inventoryNumber = "inventory-$field",
                description = "Some field device"
            )
        )
```

```
}

@Component
class MemoryUserRepository: UserRepository {
   override fun findUsers(fieldId: String, from: LocalDate, to: LocalDate): List<User> =
      listOf(
         User("1", "John", "Smith"),
         User("2", "Bryan", "Turner")
      )
}

@Component
class MemoryMeasurementRepository: MeasurementRepository {
   override fun getByDeviceIdAndPeriod(deviceId: String, fieldId: String, from: LocalDate, to: LocalDate):
List<Measurement> =
      listOf(
         Measurement(1L, LocalDateTime.now(), createList(), Location(20.0, 21.0), "type1", "field-id-1",
deviceId),
         Measurement(2L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "type2",
"field-id-1", deviceId),
         Measurement(225L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "type2",
"field-id-1", deviceId),
         Measurement(352L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "type2",
"field-id-1", deviceId),
         Measurement(452L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "type2",
"field-id-1", deviceId),
         Measurement(6582L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "type2",
"field-id-1", deviceId),
         Measurement(4582L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "type2",
"field-id-1", deviceId),
         Measurement(5842L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "type2",
"field-id-1", deviceId),
         Measurement(46542L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "last",
"field-id-1", deviceId),
         Measurement(20L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "type2",
"field-id-1", deviceId),
         Measurement(0L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "first", "field-
id-1", deviceId),
         Measurement(2L, LocalDateTime.now().plusDays(2), createList(), Location(21.0, 22.0), "type2",
"field-id-1", deviceId),
         Measurement(3L, LocalDateTime.now().plusDays(4), createList(), Location(23.0, 24.0), "type3",
"field-id-1", deviceId),
         Measurement(4L, LocalDateTime.now().plusDays(6), createList(), Location(25.0, 26.0), "type4",
"field-id-1", deviceId)
      )
}

private fun createList() = List(19){5000 * Random.nextDouble()}

package agriculture.domain.entity

data class Company(
```

```
    val id: String,
    val name: String
)

package agriculture.domain.entity

data class Device(
    val id: String,
    val code: String,
    val inventoryNumber: String?,
    val description: String? = null
)

package agriculture.domain.entity

data class Field(
    val id: String,
    val name: String,
    val square: Double,
    val groups: List<FieldGroup>,
    val coordinates: List<Location>
)

data class FieldGroup(
    val id: String,
    val name: String
)

package agriculture.domain.entity

data class Location(
        val latitude: Double,
        val longitude: Double
)

package agriculture.domain.entity

import java.time.LocalDateTime

data class Measurement(
    var id: Long,
    val dateTime: LocalDateTime,
    val densities: List<Double>,
    val location: Location,
    val tipType: String,
    val fieldId: String,
    val deviceId: String
)

package agriculture.domain.entity

data class User(
```

```
    val id: String,
    val firstName: String,
    val lastName: String
)

package agriculture.domain.interactor

import agriculture.domain.FatDevice
import agriculture.domain.InputReportRequest
import agriculture.domain.PreparedReportData
import agriculture.domain.entity.Device
import agriculture.domain.entity.Measurement
import agriculture.domain.repository.*
import agriculture.utils.formatDate
import agriculture.utils.parseDate
import org.springframework.stereotype.Component
import java.time.LocalDate

const val MEASURES_NUMBER = 19
const val DEPTH_POINT_STEP = 2.5

@Component
class PrepareReportDataInteractor(
        private val fieldRepository: FieldRepository,
        private val deviceRepository: DeviceRepository,
        private val measurementRepository: MeasurementRepository,
        private val userRepository: UserRepository,
        private val companyRepository: CompanyRepository
) {
    fun prepare(request: InputReportRequest): PreparedReportData {

        val field = fieldRepository.getById(request.fieldId)
        val devices = getDevices(request)

        val fatDevices = devices.map { mapToFatDevice(it, field.id, request.from, request.to) }

        val companyName = companyRepository.getByFieldId(field.id).name
        val dateFrom = formatDate(request.from)
        val dateTo = formatDate(request.to)

        return PreparedReportData(
                field = field,
                devices = fatDevices,
                depthMeasurePoints = getDepthPoints(),
                unit = request.unit,
                companyName = companyName,
                dateFrom = parseDate(dateFrom),
                dateTo = parseDate(dateTo)
        )
    }

    private fun getDevices(request: InputReportRequest) =
```

```
        if (request.deviceId == null)
            deviceRepository.getByFieldId(request.fieldId, request.from, request.to)
        else listOf(deviceRepository.getByDeviceId(request.deviceId))

    private fun mapToFatDevice(
        device: Device, fieldId: String, from: LocalDate, to: LocalDate
    ) = FatDevice(
        device,
        getResponsibleUsers(device.id, from, to),
        getMeasurements(device.id, fieldId, from, to)
    )

    private fun getMeasurements(
        deviceId: String, fieldId: String, from: LocalDate, to: LocalDate
    ): List<Measurement> =
        measurementRepository.getByDeviceIdAndPeriod(deviceId, fieldId, from, to)
            .distinctBy { it.id }

    private fun getResponsibleUsers(
        deviceId: String, from: LocalDate, to: LocalDate
    ) = userRepository.findUsers(deviceId, from, to).distinct()

    private fun getDepthPoints() =
        List(MEASURES_NUMBER) { it * DEPTH_POINT_STEP }

}

package agriculture.domain.repository

import agriculture.domain.entity.Company

interface CompanyRepository {
    fun getByFieldId(fieldId: String): Company
}

package agriculture.domain.repository

import agriculture.domain.entity.Device
import java.time.LocalDate

interface DeviceRepository {
    fun getByFieldId(field: String, dateFrom: LocalDate, dateTo: LocalDate): List<Device>
    fun getByDeviceId(deviceId: String): Device
}

package agriculture.domain.repository

import agriculture.domain.entity.Field

interface FieldRepository {
    fun getById(fieldId: String): Field
}
```

```
package agriculture.domain.repository

import agriculture.domain.entity.Measurement
import java.time.LocalDate
import java.time.OffsetDateTime

interface MeasurementRepository {
    fun getByDeviceIdAndPeriod(
        deviceId: String,
        fieldId: String,
        from: LocalDate,
        to: LocalDate
    ): List<Measurement>
}

package agriculture.domain.repository

import agriculture.domain.entity.User
import java.time.LocalDate

interface UserRepository{
    fun findUsers(
        deviceId: String,
        from: LocalDate,
        to: LocalDate
    ): List<User>
}

package agriculture.domain

import java.time.LocalDate

data class InputReportRequest(
    val fieldId: String,
    val deviceId: String?,
    val from: LocalDate,
    val to: LocalDate,
    val unit: Unit
)

package agriculture.domain

import agriculture.domain.entity.Device
import agriculture.domain.entity.Field
import agriculture.domain.entity.Measurement
import agriculture.domain.entity.User
import java.time.LocalDate

data class PreparedReportData(
        val field: Field,
        val devices: List<FatDevice>,
        val depthMeasurePoints: List<Double>,
```

```
        val dateFrom: LocalDate,
        val dateTo: LocalDate,
        val companyName: String,
        val unit: Unit
)

data class FatDevice(
    val device: Device,
    val responsibleUsers: List<User>,
    val measurements: List<Measurement>
)

enum class Unit(val value: String) {
    KPA("kPa"), PSI("pSi"), OTHER("other")
}
```

```
package agriculture.presentation.document.excel;

import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFCellStyle;
import org.apache.poi.xssf.usermodel.XSSFColor;
import java.awt.Color;

public class CellStyleProvider {

    CellStyle bold = null;
    CellStyle topBottomRight = null;
    CellStyle topBottomRightBold = null;

    CellStyle center = null;

    void setCellStyles(Workbook workbook) {

        //font size 10
        Font font = workbook.createFont();
        font.setFontHeightInPoints((short) 10);

        //Bold Fond
        Font boldFont = workbook.createFont();
        boldFont.setBold(true);
        boldFont.setFontHeightInPoints((short) 10);

        //Bold style
        this.bold = createCellStyle(workbook, boldFont);
        this.bold.setBorderBottom(BorderStyle.NONE);

        //Setup style for Top/Bottom/Right corner cell Border Lines
        topBottomRight = createCellStyle(workbook, font);
        applyBottomBorderStyle(topBottomRight);
        applyTopBorderStyle(topBottomRight);
        applyRightBorderStyle(topBottomRight);
```

```
    topBottomRightBold = createCellStyle(workbook, boldFont);
    applyBottomBorderStyle(topBottomRightBold);
    applyTopBorderStyle(topBottomRightBold);
    applyRightBorderStyle(topBottomRightBold);

    center = workbook.createCellStyle();
    center.setAlignment(HorizontalAlignment.CENTER);
    center.setFont(boldFont);

  }

  private CellStyle createCellStyle(Workbook workbook, Font font) {
    CellStyle cellStyle = workbook.createCellStyle();
    cellStyle.setWrapText(true);
    cellStyle.setFont(font);
    return cellStyle;
  }

  private void applyTopBorderStyle(CellStyle cellStyle) {
    cellStyle.setBorderTop(BorderStyle.THIN);
    cellStyle.setTopBorderColor(IndexedColors.BLACK.getIndex());
  }

  private void applyBottomBorderStyle(CellStyle cellStyle) {
    cellStyle.setBorderBottom(BorderStyle.THIN);
    cellStyle.setBottomBorderColor(IndexedColors.BLACK.getIndex());
  }

  private void applyLeftBorderStyle(CellStyle cellStyle) {
    cellStyle.setBorderLeft(BorderStyle.THIN);
    cellStyle.setLeftBorderColor(IndexedColors.BLACK.getIndex());
  }

  private void applyRightBorderStyle(CellStyle cellStyle) {
    cellStyle.setBorderRight(BorderStyle.THIN);
    cellStyle.setRightBorderColor(IndexedColors.BLACK.getIndex());
  }

  private CellStyle createColorCellStyle(Workbook workbook, java.awt.Color color) {
    CellStyle cellStyle = workbook.createCellStyle();

    XSSFColor xssfColor = new XSSFColor(color);
    ((XSSFCellStyle) cellStyle).setFillForegroundColor(xssfColor);
    cellStyle.setFillPattern(FillPatternType.SOLID_FOREGROUND);

    applyBottomBorderStyle(cellStyle);
    applyRightBorderStyle(cellStyle);
    applyTopBorderStyle(cellStyle);
    applyLeftBorderStyle(cellStyle);

    return cellStyle;
  }
```

```
    public CellStyle createColorCellStyle(Workbook workbook, agriculture.presentation.document.Color
color) {
        Color awtColor = new Color(color.getRed(), color.getGreen(), color.getBlue());
        return createColorCellStyle(workbook, awtColor);
    }

}

package agriculture.presentation.document.excel;

import agriculture.presentation.document.ChartData;
import agriculture.presentation.document.MultiChartData;
import agriculture.presentation.document.Series;
import org.apache.poi.xddf.usermodel.chart.*;
import org.apache.poi.xssf.usermodel.XSSFChart;
import org.apache.poi.xssf.usermodel.XSSFClientAnchor;
import org.apache.poi.xssf.usermodel.XSSFDrawing;
import org.apache.poi.xssf.usermodel.XSSFSheet;

import java.util.Collections;
import java.util.List;

public class ChartCreator {

    void insertSingleLineChart(
            XSSFSheet sheet,
            int columnIndex, int width,
            int rowIndex, int height,
            ChartData chartData
    ) {
        List<Series> oneSeriesList = Collections.singletonList(chartData.getDensities());
        insertChart(sheet,
                columnIndex, width,
                rowIndex, height,
                chartData.getXTitle(), chartData.getYTitle(),
                chartData.getDepthArguments(), oneSeriesList,
                chartData.getTitle(),
                false);
    }

    void insertMultiLineChart(
            XSSFSheet sheet,
            int columnIndex, int width,
            int rowIndex, int height,
            MultiChartData chartData
    ) {
        insertChart(sheet,
                columnIndex, width,
                rowIndex, height,
                chartData.getXTitle(), chartData.getYTitle(),
                chartData.getDepthArguments(), chartData.getDensitySeries(),
```

```
            chartData.getTitle(),
            true);
    }

    private void insertChart(
            XSSFSheet sheet,
            int columnIndex, int width,
            int rowIndex, int height,
            String xAxisTitle,
            String yAxisTitle,
            String[] arguments,
            List<Series> values,
            String title,
            boolean showLegend
    ) {
        XSSFDrawing drawing = sheet.createDrawingPatriarch();
        XSSFClientAnchor anchor = drawing.createAnchor(0, 0, 0, 0, columnIndex, rowIndex, columnIndex +
width, rowIndex + height);

        XSSFChart chart = drawing.createChart(anchor);

        if (showLegend) {
            XDDFChartLegend legend = chart.getOrAddLegend();
            legend.setPosition(LegendPosition.BOTTOM);
        }

        // Use a category axis for the bottom axis.
        XDDFCategoryAxis bottomAxis = chart.createCategoryAxis(AxisPosition.BOTTOM);
        bottomAxis.setTitle(xAxisTitle); // https://stackoverflow.com/questions/32010765
        XDDFValueAxis leftAxis = chart.createValueAxis(AxisPosition.LEFT);
        leftAxis.setTitle(yAxisTitle);
        leftAxis.setCrosses(AxisCrosses.AUTO_ZERO);

        XDDFLineChartData data = (XDDFLineChartData) chart.createData(ChartTypes.LINE, bottomAxis,
leftAxis);
        XDDFDataSource<String> xs = XDDFDataSourcesFactory.fromArray(arguments, "");

        for (Series series: values) {
            XDDFNumericalDataSource<Double> ys1 = XDDFDataSourcesFactory.fromArray(series.getValues(),
"");
            XDDFLineChartData.Series series1 = (XDDFLineChartData.Series) data.addSeries(xs, ys1);
            series1.setTitle(series.getTitle(), null); // https://stackoverflow.com/questions/21855842
        }

        chart.plot(data);
        chart.setTitleText(title);

        //if your series have missing values like https://stackoverflow.com/questions/29014848
        chart.displayBlanksAs(DisplayBlanks.GAP);
    }

}
```

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ О.А. Павлов
“___” _____ 2019 р.

Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства

Графічні матеріали

КПІ.ІП-5103.045440.07.99

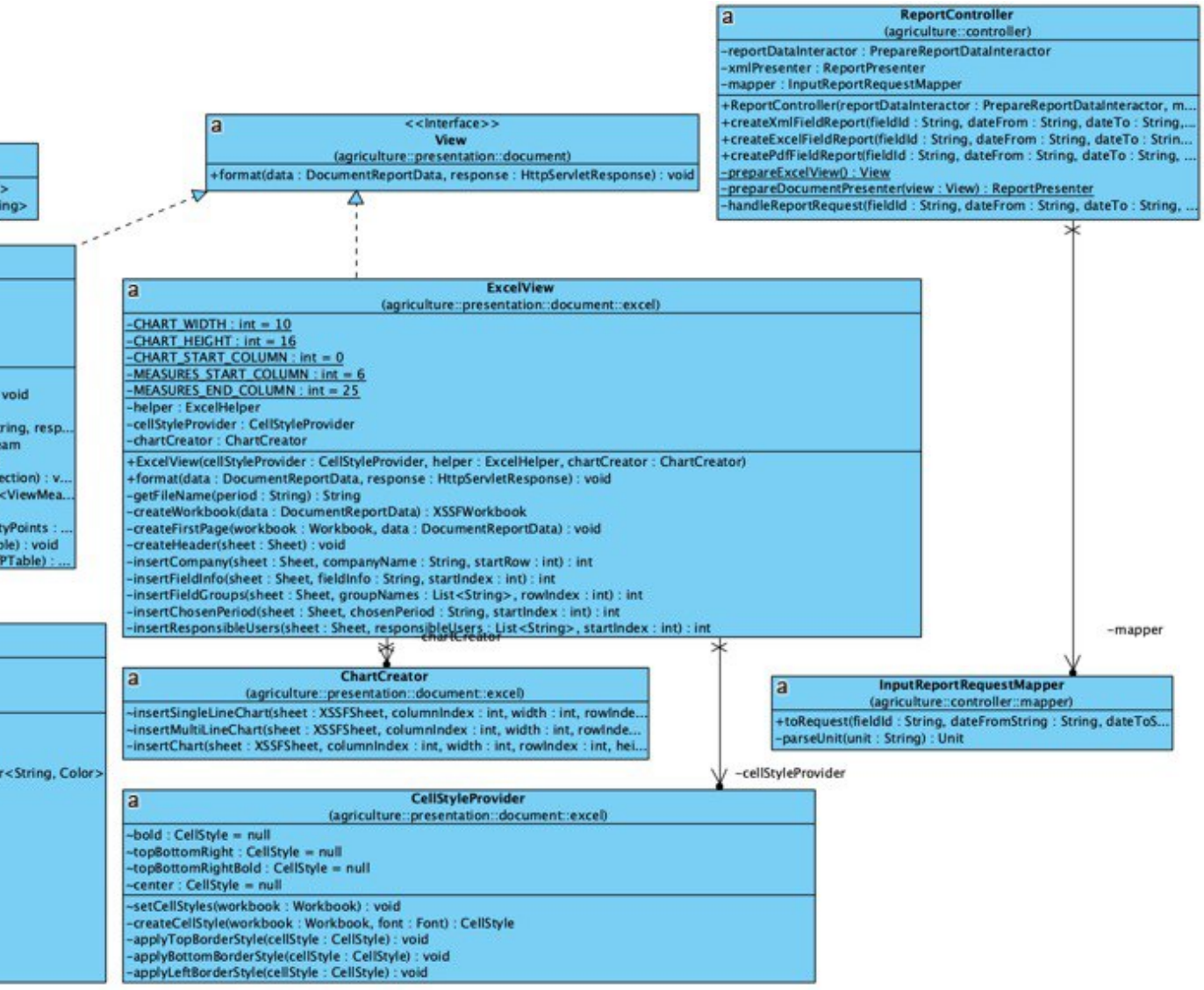
“ПОГОДЖЕНО”
Керівник проекту:
_____ Д.С. Смаковський

Нормоконтроль:
_____М.М. Головченко

Виконавець:
_____М.В. Гарбовський

Київ – 2019 року

КПІ.ІП-5103.045440.07.99.СС



					КПІ.ІП-5103.045440.07.99.СС	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Гарбовський М.В.				
Перевірив		Смаковський Д.С.				Аркуш / Аркушів
Т. кон.						
					Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51
Н. кон.		Головченко М.М.				
Затвердив		Смаковський Д.С.				

Тип грунту:	Пухкий			Щільний			Переущільнений		
Щільність	0-950	950-1400	1400-1750	1750-1900	1900-2300	2300-2900	2900-3500	3500-4000	40

№	1	2	3	4	5	6	7	8	9	
Дата	23.05.19	21.05.19	23.05.19	25.05.19	27.05.19	23.05.19	23.05.19	23.05.19	23.05.19	2
Час	03:42:09	03:42:09	03:42:09	03:42:09	03:42:09	03:42:09	03:42:09	03:42:09	03:42:09	0
ID пристрою	device-field_id_1	device-field_id_1	device-field_id_1	device-field_id_1	device-field_id_1	device-field_id_1	device-field_id_1	device-field_id_1	device-field_id_1	devic
Координати	Google map	Google map	Google map	Google map	Google map	Google map	Google map	Google map	Google map	Go
Наконечник	first	type1	type2	type3	type4	type2	type2	type2	type2	
Глибина(см)	Щільність									
0.0	4030	2574	3313	2905	121	2513	1743	2593	1568	
2.5	4461	4162	29	3386	40	1775	1867	705	4973	
5.0	779	4173	3075	2931	3360	4203	3344	1710	916	
7.5	4528	3196	877	121	2791	382	4972	3061	3217	
10.0	3756	2327	121	2363	2090	1011	3665	3935	1611	
12.5	3250	1813	3640	3908	348	3869	3697	3426	4053	
15.0	4801	3673	1403	3337	477	1482	821	4226	1784	
17.5	399	2725	3682	4569	3829	1389	3689	3324	4737	
20.0	4127	930	2323	3307	4833	50	1476	3330	839	
22.5	1611	3838	2112	4903	2759	1087	3223	932	4027	
25.0	3762	1632	2844	1529	854	3257	1528	1131	3526	
27.5	4637	3013	3058	2000	1989	221	2219	1060	1024	
30.0	2130	737	1023	4971	2135	886	586	2918	3129	
32.5	3576	451	1898	944	2493	2268	4919	3493	2816	
35.0	1656	4258	302	87	3489	1447	2463	4365	3529	
37.5	2171	4184	332	663	4725	3177	1368	4275	4390	
40.0	1805	1321	4771	4166	2970	2249	4596	4212	3754	
42.5	945	670	1146	2257	3237	3735	1256	1247	1284	
45.0	816	1750	2406	1349	488	664	86	3603	456	

Компанія:AgriCompany field_id_1
Поле field-name-field_id_1 (Площа 0.06 га)
Група полів, якої належить обране поле
field-group-1
field-group-2
field-group-3
field-group-4
Обраний період: 12.12.12-14.12.12

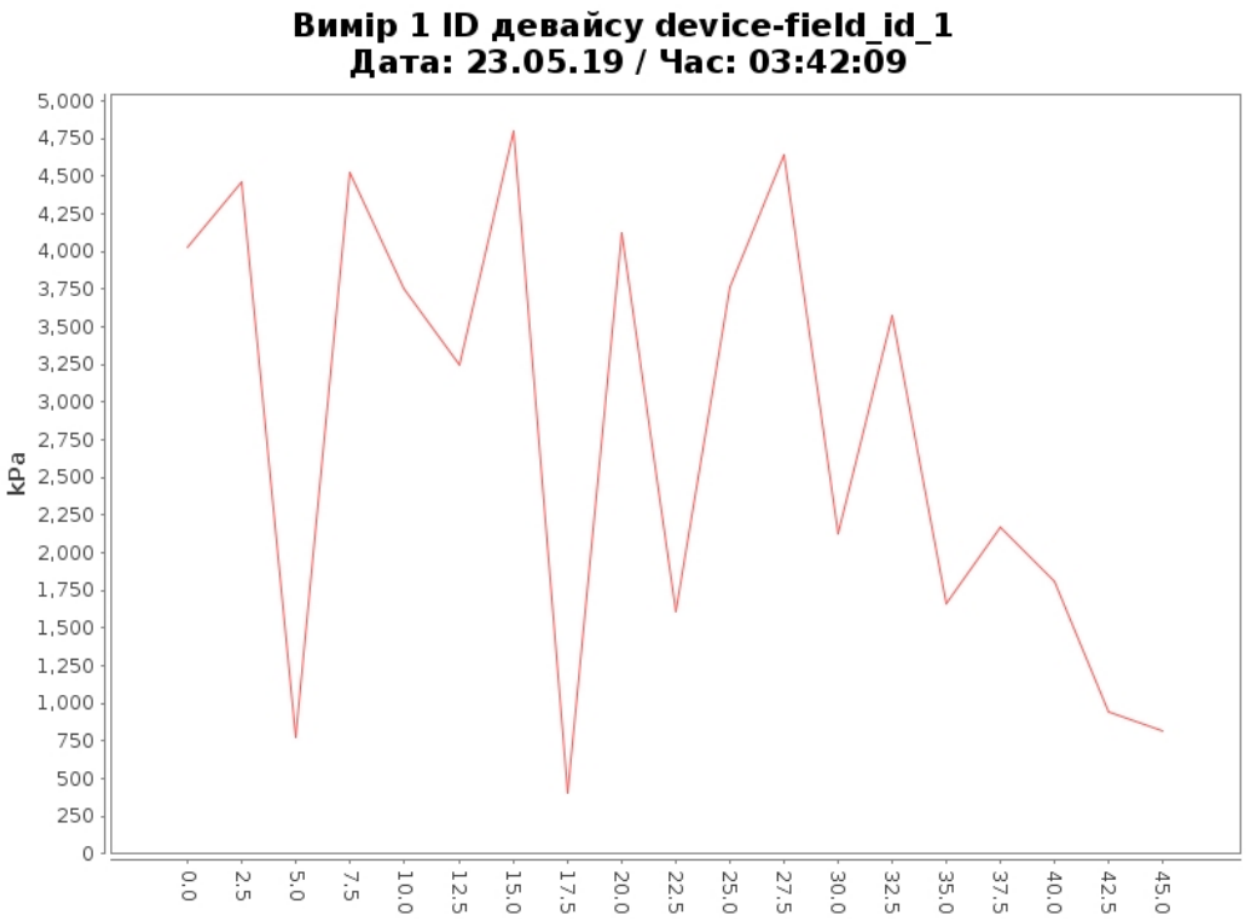
Відповідальні особи за обраний період:
John-Smith
Bryan-Turner

		Тип грунту:	Пухкий			Щільний			Переущільнений		
	kPa	Щільність	0-950	950-1400	1400-1750	1750-1900	1900-2300	2300-2900	2900-3500	3500-4000	4000-6000

№	ID пристрою	Координати	Дата	Час	Наконечник	0.0	2.5	5.0	7.5	10.0	12.5	15.0	17.5	20.0
1	device-field_id_1	Google map	23.05.19	03:39:54	first	1762	1560	4070	980	3375	1798	3298	1371	660
2	device-field_id_1	Google map	21.05.19	03:39:54	type1	409	1890	3164	1615	3303	3980	4347	4628	314
3	device-field_id_1	Google map	23.05.19	03:39:54	type2	238	3244	4143	55	3136	710	351	3233	657
4	device-field_id_1	Google map	25.05.19	03:39:54	type3	4183	1926	1049	572	1073	1463	3344	4145	3695
5	device-field_id_1	Google map	27.05.19	03:39:54	type4	1574	3568	1020	3442	1556	4916	718	1105	1231
6	device-field_id_1	Google map	23.05.19	03:39:54	type2	1991	4648	1571	953	3246	2210	3692	2839	818
7	device-field_id_1	Google map	23.05.19	03:39:54	type2	2759	458	473	3361	686	3179	2431	1771	1491
8	device-field_id_1	Google map	23.05.19	03:39:54	type2	3120	312	2290	3832	3161	1917	3872	1487	1607
9	device-field_id_1	Google map	23.05.19	03:39:54	type2	3810	1174	4962	290	317	2398	182	2373	1517
10	device-field_id_1	Google map	23.05.19	03:39:54	type2	1413	4235	4706	865	423	627	3857	964	1679
11	device-field_id_1	Google map	23.05.19	03:39:54	type2	3811	972	2170	2883	1734	2817	2158	1908	1675

00-6000

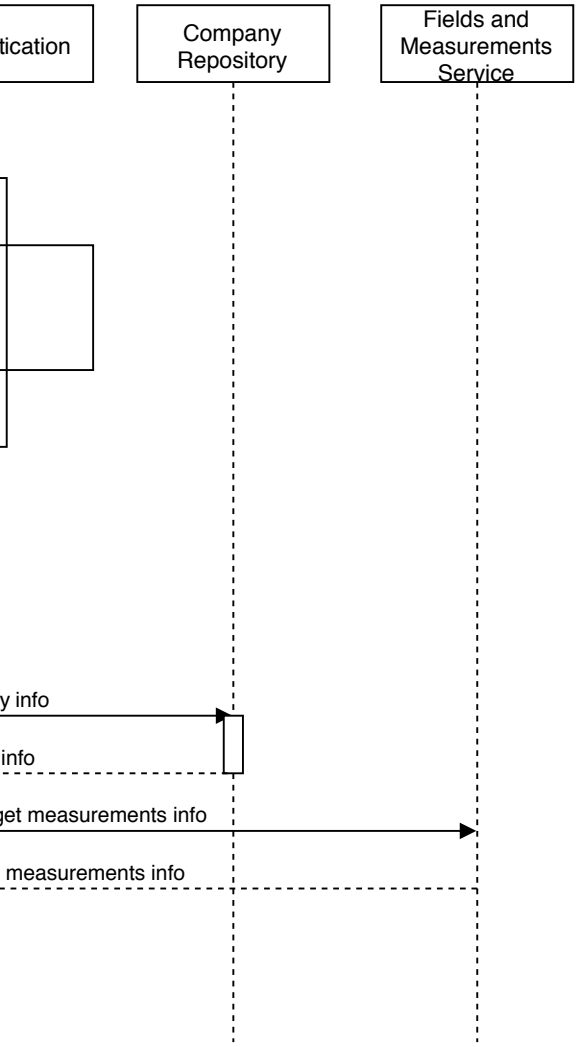
10
23.05.19
03:42:09
e-field_id_1
oogle map
type2
1909
1979
49
3506
4115
2661
3915
2438
2782
1162
1472
4447
679
3179
3970
4772
2914
2757
1682



22.5
1639
1330
4724
4192
2863
1937
1132
3188
3401
4525
1713

					КПІ.ІП-5103.045440.07.99.КЗ	
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду звітних форм	Літера / Маса / Масштаб
Розробив		Гарбовський М.В.				
Перевірив		Смаковський Д.С.				Аркуш / Аркушів
Т. кон.						
					Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51
Н. кон.		Ліщук К.І.				
Затвердив		Смаковський Д.С.				

КПІ.ІП-5103.045440.07.99.СС



					КПІ.ІП-5103.045440.07.99.СС		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна діяльності	Літера	Маса Масштаб
Розробив							
Перевірив							
Т. кон.						Аркуш	Аркушів
					Мікросервіс генерації репортів хмарної мікросервісної геоінформаційної системи для сільського господарства	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51	
Н. кон.							
Затвердив		Смаковський Д.С.					